TLDR Pages

The Book

Simplified and community-driven man pages

[tldr-pages.github.io](https://tldr-pages.github.io)
## Contents

1 COMMON

- 7z ................................................................. 16
- 7za ............................................................... 16
- 7zr ............................................................... 17
- ab ................................................................. 17
- ack ............................................................... 18
- adb ............................................................... 18
- ag ................................................................. 19
- alias ............................................................. 19
- ansible-playbook ............................................. 20
- ansible .......................................................... 20
- apc ............................................................... 21
- apm ............................................................... 21
- apropos ........................................................... 22
- ar ................................................................. 22
- aria2c ............................................................ 23
- arp ............................................................... 23
- assimp ........................................................... 24
- atom ............................................................. 24
- autojump ........................................................ 25
- autossh .......................................................... 25
- avrdude .......................................................... 26
- awk ............................................................... 26
- aws s3 ............................................................ 27
- axel ............................................................. 27
- base32 ........................................................... 28
- base64 ........................................................... 28
- basename ........................................................ 28
- bash .............................................................. 29
- bashmarks ....................................................... 29
- bc ................................................................. 30
- bedtools ........................................................ 30
- bg ................................................................. 31
- blender .......................................................... 31
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>dhcpwn</td>
<td>52</td>
</tr>
<tr>
<td>diff</td>
<td>52</td>
</tr>
<tr>
<td>dig</td>
<td>53</td>
</tr>
<tr>
<td>dirs</td>
<td>53</td>
</tr>
<tr>
<td>docker</td>
<td>54</td>
</tr>
<tr>
<td>dokku</td>
<td>54</td>
</tr>
<tr>
<td>drush</td>
<td>55</td>
</tr>
<tr>
<td>ebook-convert</td>
<td>55</td>
</tr>
<tr>
<td>echo</td>
<td>55</td>
</tr>
<tr>
<td>ed</td>
<td>56</td>
</tr>
<tr>
<td>electrum</td>
<td>57</td>
</tr>
<tr>
<td>emacs</td>
<td>57</td>
</tr>
<tr>
<td>enca</td>
<td>58</td>
</tr>
<tr>
<td>env</td>
<td>58</td>
</tr>
<tr>
<td>espeak</td>
<td>59</td>
</tr>
<tr>
<td>exiftool</td>
<td>59</td>
</tr>
<tr>
<td>fdupes</td>
<td>60</td>
</tr>
<tr>
<td>ffmpeg</td>
<td>60</td>
</tr>
<tr>
<td>fg</td>
<td>60</td>
</tr>
<tr>
<td>file</td>
<td>61</td>
</tr>
<tr>
<td>find</td>
<td>61</td>
</tr>
<tr>
<td>fold</td>
<td>62</td>
</tr>
<tr>
<td>for</td>
<td>62</td>
</tr>
<tr>
<td>fortune</td>
<td>62</td>
</tr>
<tr>
<td>fsck</td>
<td>63</td>
</tr>
<tr>
<td>fswatch</td>
<td>63</td>
</tr>
<tr>
<td>fswebcam</td>
<td>64</td>
</tr>
<tr>
<td>ftp</td>
<td>64</td>
</tr>
<tr>
<td>fuck</td>
<td>65</td>
</tr>
<tr>
<td>fzf</td>
<td>65</td>
</tr>
<tr>
<td>gcc</td>
<td>66</td>
</tr>
<tr>
<td>gdb</td>
<td>66</td>
</tr>
<tr>
<td>gem</td>
<td>67</td>
</tr>
<tr>
<td>ghc</td>
<td>67</td>
</tr>
<tr>
<td>gifsicle</td>
<td>68</td>
</tr>
<tr>
<td>git</td>
<td>68</td>
</tr>
<tr>
<td>git add</td>
<td>68</td>
</tr>
<tr>
<td>git bisect</td>
<td>69</td>
</tr>
<tr>
<td>git blame</td>
<td>69</td>
</tr>
<tr>
<td>git branch</td>
<td>70</td>
</tr>
<tr>
<td>git checkout</td>
<td>70</td>
</tr>
<tr>
<td>git cherry-pick</td>
<td>71</td>
</tr>
<tr>
<td>git clone</td>
<td>71</td>
</tr>
<tr>
<td>git commit</td>
<td>71</td>
</tr>
<tr>
<td>git config</td>
<td>72</td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
</tr>
<tr>
<td>in2csv</td>
<td>93</td>
</tr>
<tr>
<td>inkscape</td>
<td>94</td>
</tr>
<tr>
<td>install</td>
<td>94</td>
</tr>
<tr>
<td>ionice</td>
<td>95</td>
</tr>
<tr>
<td>ioping</td>
<td>95</td>
</tr>
<tr>
<td>ipcs</td>
<td>95</td>
</tr>
<tr>
<td>jar</td>
<td>96</td>
</tr>
<tr>
<td>java</td>
<td>96</td>
</tr>
<tr>
<td>javad</td>
<td>96</td>
</tr>
<tr>
<td>jHat</td>
<td>97</td>
</tr>
<tr>
<td>jmap</td>
<td>97</td>
</tr>
<tr>
<td>jobs</td>
<td>97</td>
</tr>
<tr>
<td>jd</td>
<td>98</td>
</tr>
<tr>
<td>jstack</td>
<td>98</td>
</tr>
<tr>
<td>julia</td>
<td>99</td>
</tr>
<tr>
<td>kill</td>
<td>99</td>
</tr>
<tr>
<td>last</td>
<td>100</td>
</tr>
<tr>
<td>latexmk</td>
<td>101</td>
</tr>
<tr>
<td>less</td>
<td>101</td>
</tr>
<tr>
<td>license</td>
<td>102</td>
</tr>
<tr>
<td>ln</td>
<td>102</td>
</tr>
<tr>
<td>logstash</td>
<td>103</td>
</tr>
<tr>
<td>ld</td>
<td>103</td>
</tr>
<tr>
<td>lpsstat</td>
<td>104</td>
</tr>
<tr>
<td>ls</td>
<td>104</td>
</tr>
<tr>
<td>lsof</td>
<td>105</td>
</tr>
<tr>
<td>lwp-request</td>
<td>105</td>
</tr>
<tr>
<td>mailx</td>
<td>106</td>
</tr>
<tr>
<td>make</td>
<td>106</td>
</tr>
<tr>
<td>man</td>
<td>107</td>
</tr>
<tr>
<td>mdp</td>
<td>107</td>
</tr>
<tr>
<td>meshlabserver</td>
<td>108</td>
</tr>
<tr>
<td>meteor</td>
<td>108</td>
</tr>
<tr>
<td>mitmdump</td>
<td>109</td>
</tr>
<tr>
<td>mitmproxy</td>
<td>109</td>
</tr>
<tr>
<td>mkdir</td>
<td>109</td>
</tr>
<tr>
<td>mkfifo</td>
<td>110</td>
</tr>
<tr>
<td>mmv</td>
<td>110</td>
</tr>
<tr>
<td>mocha</td>
<td>110</td>
</tr>
<tr>
<td>mongo</td>
<td>110</td>
</tr>
<tr>
<td>mongodump</td>
<td>111</td>
</tr>
<tr>
<td>mongorestore</td>
<td>112</td>
</tr>
<tr>
<td>montage</td>
<td>112</td>
</tr>
<tr>
<td>more</td>
<td>113</td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>mount</td>
<td>113</td>
</tr>
<tr>
<td>mp4box</td>
<td>114</td>
</tr>
<tr>
<td>mpc</td>
<td>114</td>
</tr>
<tr>
<td>msmtop</td>
<td>115</td>
</tr>
<tr>
<td>mtr</td>
<td>115</td>
</tr>
<tr>
<td>mv</td>
<td>116</td>
</tr>
<tr>
<td>mysql</td>
<td>116</td>
</tr>
<tr>
<td>mysqldump</td>
<td>117</td>
</tr>
<tr>
<td>nano</td>
<td>117</td>
</tr>
<tr>
<td>nc</td>
<td>117</td>
</tr>
<tr>
<td>nginx</td>
<td>118</td>
</tr>
<tr>
<td>ngrep</td>
<td>118</td>
</tr>
<tr>
<td>nice</td>
<td>119</td>
</tr>
<tr>
<td>nix-env</td>
<td>119</td>
</tr>
<tr>
<td>nmap</td>
<td>120</td>
</tr>
<tr>
<td>node</td>
<td>120</td>
</tr>
<tr>
<td>nohup</td>
<td>120</td>
</tr>
<tr>
<td>npm</td>
<td>120</td>
</tr>
<tr>
<td>nslookup</td>
<td>121</td>
</tr>
<tr>
<td>nvm</td>
<td>121</td>
</tr>
<tr>
<td>odps auth</td>
<td>122</td>
</tr>
<tr>
<td>odps func</td>
<td>122</td>
</tr>
<tr>
<td>odps inst</td>
<td>122</td>
</tr>
<tr>
<td>odps resource</td>
<td>122</td>
</tr>
<tr>
<td>odps table</td>
<td>122</td>
</tr>
<tr>
<td>odps tunnel</td>
<td>122</td>
</tr>
<tr>
<td>openssl</td>
<td>123</td>
</tr>
<tr>
<td>optipng</td>
<td>123</td>
</tr>
<tr>
<td>parallel</td>
<td>124</td>
</tr>
<tr>
<td>passwd</td>
<td>124</td>
</tr>
<tr>
<td>patch</td>
<td>124</td>
</tr>
<tr>
<td>pdflatex</td>
<td>124</td>
</tr>
<tr>
<td>pdftk</td>
<td>124</td>
</tr>
<tr>
<td>perl</td>
<td>125</td>
</tr>
<tr>
<td>pg_dump</td>
<td>125</td>
</tr>
<tr>
<td>pg_restore</td>
<td>125</td>
</tr>
<tr>
<td>pgrep</td>
<td>125</td>
</tr>
<tr>
<td>php</td>
<td>126</td>
</tr>
<tr>
<td>ph霖ze</td>
<td>126</td>
</tr>
<tr>
<td>phounit</td>
<td>126</td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>pigz</td>
<td>135</td>
</tr>
<tr>
<td>ping</td>
<td>135</td>
</tr>
<tr>
<td>pip</td>
<td>136</td>
</tr>
<tr>
<td>pkill</td>
<td>136</td>
</tr>
<tr>
<td>play</td>
<td>137</td>
</tr>
<tr>
<td>pngcrush</td>
<td>137</td>
</tr>
<tr>
<td>printf</td>
<td>138</td>
</tr>
<tr>
<td>ps</td>
<td>138</td>
</tr>
<tr>
<td>psql</td>
<td>139</td>
</tr>
<tr>
<td>ps</td>
<td>139</td>
</tr>
<tr>
<td>pushd</td>
<td>139</td>
</tr>
<tr>
<td>py</td>
<td>140</td>
</tr>
<tr>
<td>pwd</td>
<td>140</td>
</tr>
<tr>
<td>pyenv</td>
<td>141</td>
</tr>
<tr>
<td>python</td>
<td>141</td>
</tr>
<tr>
<td>q</td>
<td>142</td>
</tr>
<tr>
<td>qemu-img</td>
<td>142</td>
</tr>
<tr>
<td>qemu</td>
<td>143</td>
</tr>
<tr>
<td>quota</td>
<td>143</td>
</tr>
<tr>
<td>read</td>
<td>144</td>
</tr>
<tr>
<td>readlink</td>
<td>144</td>
</tr>
<tr>
<td>redis-cli</td>
<td>145</td>
</tr>
<tr>
<td>redshift</td>
<td>145</td>
</tr>
<tr>
<td>rename</td>
<td>146</td>
</tr>
<tr>
<td>renice</td>
<td>146</td>
</tr>
<tr>
<td>rev</td>
<td>146</td>
</tr>
<tr>
<td>ripgrep</td>
<td>147</td>
</tr>
<tr>
<td>rm</td>
<td>148</td>
</tr>
<tr>
<td>rmdir</td>
<td>148</td>
</tr>
<tr>
<td>route</td>
<td>148</td>
</tr>
<tr>
<td>rsync</td>
<td>148</td>
</tr>
<tr>
<td>rtv</td>
<td>149</td>
</tr>
<tr>
<td>rustc</td>
<td>149</td>
</tr>
<tr>
<td>sails</td>
<td>150</td>
</tr>
<tr>
<td>salt-key</td>
<td>150</td>
</tr>
<tr>
<td>salt-key</td>
<td>151</td>
</tr>
<tr>
<td>salt-run</td>
<td>152</td>
</tr>
<tr>
<td>salt</td>
<td>152</td>
</tr>
<tr>
<td>samtools</td>
<td>152</td>
</tr>
<tr>
<td>sass</td>
<td>152</td>
</tr>
<tr>
<td>scp</td>
<td>153</td>
</tr>
<tr>
<td>scrapy</td>
<td>154</td>
</tr>
<tr>
<td>screen</td>
<td>154</td>
</tr>
<tr>
<td>screenfetch</td>
<td>155</td>
</tr>
<tr>
<td>sed</td>
<td>156</td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
</tr>
<tr>
<td>sendmail</td>
<td>156</td>
</tr>
<tr>
<td>sed</td>
<td>156</td>
</tr>
<tr>
<td>sftp</td>
<td>157</td>
</tr>
<tr>
<td>ssh</td>
<td>158</td>
</tr>
<tr>
<td>ssh-copy-id</td>
<td>160</td>
</tr>
<tr>
<td>ssh-copy-id</td>
<td>163</td>
</tr>
<tr>
<td>ssh-keygen</td>
<td>168</td>
</tr>
<tr>
<td>sshfs</td>
<td>164</td>
</tr>
<tr>
<td>st-flash</td>
<td>166</td>
</tr>
<tr>
<td>st-info</td>
<td>166</td>
</tr>
<tr>
<td>st-util</td>
<td>168</td>
</tr>
<tr>
<td>strings</td>
<td>169</td>
</tr>
<tr>
<td>su</td>
<td>167</td>
</tr>
<tr>
<td>sudo</td>
<td>168</td>
</tr>
<tr>
<td>sum</td>
<td>168</td>
</tr>
<tr>
<td>supervisorctl</td>
<td>168</td>
</tr>
<tr>
<td>supervisord</td>
<td>169</td>
</tr>
<tr>
<td>svn</td>
<td>169</td>
</tr>
<tr>
<td>tabula</td>
<td>170</td>
</tr>
<tr>
<td>tac</td>
<td>170</td>
</tr>
<tr>
<td>tail</td>
<td>171</td>
</tr>
<tr>
<td>tar</td>
<td>171</td>
</tr>
<tr>
<td>task</td>
<td>172</td>
</tr>
<tr>
<td>tcpdump</td>
<td>172</td>
</tr>
<tr>
<td>tee</td>
<td>173</td>
</tr>
<tr>
<td>telnet</td>
<td>173</td>
</tr>
<tr>
<td>tesseract</td>
<td>173</td>
</tr>
<tr>
<td>test</td>
<td>174</td>
</tr>
<tr>
<td>time</td>
<td>174</td>
</tr>
<tr>
<td>tldr</td>
<td>175</td>
</tr>
<tr>
<td>tidir</td>
<td>175</td>
</tr>
<tr>
<td>tmux</td>
<td>175</td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
</tr>
<tr>
<td>touch</td>
<td>176</td>
</tr>
<tr>
<td>tput</td>
<td>176</td>
</tr>
<tr>
<td>tr</td>
<td>177</td>
</tr>
<tr>
<td>traceroute</td>
<td>177</td>
</tr>
<tr>
<td>transcode</td>
<td>178</td>
</tr>
<tr>
<td>tty</td>
<td>178</td>
</tr>
<tr>
<td>ufraw-batch</td>
<td>178</td>
</tr>
<tr>
<td>umount</td>
<td>178</td>
</tr>
<tr>
<td>unrar</td>
<td>179</td>
</tr>
<tr>
<td>uniq</td>
<td>180</td>
</tr>
<tr>
<td>unzpipe</td>
<td>180</td>
</tr>
<tr>
<td>uptime</td>
<td>181</td>
</tr>
<tr>
<td>vagrant</td>
<td>181</td>
</tr>
<tr>
<td>vim</td>
<td>182</td>
</tr>
<tr>
<td>vmtutor</td>
<td>182</td>
</tr>
<tr>
<td>virtualenv</td>
<td>183</td>
</tr>
<tr>
<td>visudo</td>
<td>183</td>
</tr>
<tr>
<td>vue-cli</td>
<td>183</td>
</tr>
<tr>
<td>w</td>
<td>184</td>
</tr>
<tr>
<td>w3m</td>
<td>184</td>
</tr>
<tr>
<td>wait</td>
<td>184</td>
</tr>
<tr>
<td>watch</td>
<td>184</td>
</tr>
<tr>
<td>wc</td>
<td>185</td>
</tr>
<tr>
<td>webpack</td>
<td>185</td>
</tr>
<tr>
<td>wget</td>
<td>186</td>
</tr>
<tr>
<td>which</td>
<td>186</td>
</tr>
<tr>
<td>while</td>
<td>186</td>
</tr>
<tr>
<td>who</td>
<td>187</td>
</tr>
<tr>
<td>whoami</td>
<td>187</td>
</tr>
<tr>
<td>x_x</td>
<td>187</td>
</tr>
<tr>
<td>xargs</td>
<td>188</td>
</tr>
<tr>
<td>xcv</td>
<td>188</td>
</tr>
<tr>
<td>xz</td>
<td>189</td>
</tr>
<tr>
<td>yarn</td>
<td>189</td>
</tr>
<tr>
<td>yes</td>
<td>190</td>
</tr>
<tr>
<td>youtube-dl</td>
<td>190</td>
</tr>
<tr>
<td>zbarimg</td>
<td>190</td>
</tr>
<tr>
<td>zcat</td>
<td>191</td>
</tr>
<tr>
<td>zdb</td>
<td>191</td>
</tr>
<tr>
<td>zfs</td>
<td>191</td>
</tr>
<tr>
<td>zipp</td>
<td>192</td>
</tr>
<tr>
<td>zless</td>
<td>192</td>
</tr>
<tr>
<td>zpool</td>
<td>192</td>
</tr>
</tbody>
</table>
2 LINUX

adduser .................................................. 194
apt-cache .................................................. 194
apt-get ..................................................... 195
apt-key ...................................................... 195
apt ........................................................... 196
aptitude ..................................................... 196
archey ....................................................... 197
at ............................................................. 197
beep .......................................................... 198
brew .......................................................... 198
bzip2 .......................................................... 199
chattr ......................................................... 199
chroot ......................................................... 199
cmus .......................................................... 200
compose ...................................................... 200
cryptsetup .................................................. 200
date ........................................................... 201
dnfon .......................................................... 202
dpkg-query ................................................... 202
dpkg .......................................................... 203
du .............................................................. 203
edit ............................................................. 204
edquota ........................................................ 204
emerge ........................................................ 204
equery ........................................................ 205
expand ........................................................ 205
expr ............................................................. 206
fc-list ......................................................... 207
fc-match ...................................................... 207
fc-pattern .................................................... 207
figlet ........................................................... 207
findmnt ........................................................ 208
firewall-cmd .................................................. 208
free ............................................................ 208
fuser ........................................................... 209
getent ........................................................... 209
groupadd ...................................................... 210
groupdel ...................................................... 210
groupmod ..................................................... 210
halt .............................................................. 210
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>head</td>
<td>211</td>
</tr>
<tr>
<td>hostname</td>
<td>211</td>
</tr>
<tr>
<td>htop</td>
<td>212</td>
</tr>
<tr>
<td>hwclock</td>
<td>212</td>
</tr>
<tr>
<td>ifdown</td>
<td>212</td>
</tr>
<tr>
<td>ifup</td>
<td>213</td>
</tr>
<tr>
<td>iostat</td>
<td>213</td>
</tr>
<tr>
<td>ip</td>
<td>214</td>
</tr>
<tr>
<td>iptables</td>
<td>214</td>
</tr>
<tr>
<td>jobs</td>
<td>215</td>
</tr>
<tr>
<td>journalctl</td>
<td>215</td>
</tr>
<tr>
<td>locate</td>
<td>216</td>
</tr>
<tr>
<td>logger</td>
<td>216</td>
</tr>
<tr>
<td>lsattr</td>
<td>217</td>
</tr>
<tr>
<td>lsb_release</td>
<td>217</td>
</tr>
<tr>
<td>lsblk</td>
<td>218</td>
</tr>
<tr>
<td>ltrace</td>
<td>218</td>
</tr>
<tr>
<td>md5sum</td>
<td>219</td>
</tr>
<tr>
<td>mdadm</td>
<td>219</td>
</tr>
<tr>
<td>mke2fs</td>
<td>220</td>
</tr>
<tr>
<td>mkfs.cramfs</td>
<td>220</td>
</tr>
<tr>
<td>mkfs.exfat</td>
<td>220</td>
</tr>
<tr>
<td>mkfs.fat</td>
<td>221</td>
</tr>
<tr>
<td>mkfs.minix</td>
<td>221</td>
</tr>
<tr>
<td>mkfs.ntfs</td>
<td>221</td>
</tr>
<tr>
<td>mkfs.vfat</td>
<td>222</td>
</tr>
<tr>
<td>nethogs</td>
<td>222</td>
</tr>
<tr>
<td>netstat</td>
<td>223</td>
</tr>
<tr>
<td>nm</td>
<td>223</td>
</tr>
<tr>
<td>notify-send</td>
<td>224</td>
</tr>
<tr>
<td>pacman</td>
<td>224</td>
</tr>
<tr>
<td>pkgadd</td>
<td>225</td>
</tr>
<tr>
<td>pkginfo</td>
<td>225</td>
</tr>
<tr>
<td>pkgmk</td>
<td>225</td>
</tr>
<tr>
<td>pkgrm</td>
<td>226</td>
</tr>
<tr>
<td>print</td>
<td>226</td>
</tr>
<tr>
<td>port-get</td>
<td>227</td>
</tr>
<tr>
<td>pvccreate</td>
<td>227</td>
</tr>
<tr>
<td>pwgen</td>
<td>228</td>
</tr>
<tr>
<td>quotacheck</td>
<td>228</td>
</tr>
<tr>
<td>rdesktop</td>
<td>229</td>
</tr>
<tr>
<td>reboot</td>
<td>229</td>
</tr>
<tr>
<td>repquota</td>
<td>230</td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
</tr>
<tr>
<td>rpm</td>
<td>230</td>
</tr>
<tr>
<td>run-mailcap</td>
<td>231</td>
</tr>
<tr>
<td>see</td>
<td>231</td>
</tr>
<tr>
<td>service</td>
<td>232</td>
</tr>
<tr>
<td>setfacl</td>
<td>232</td>
</tr>
<tr>
<td>sha1sum</td>
<td>233</td>
</tr>
<tr>
<td>sha224sum</td>
<td>233</td>
</tr>
<tr>
<td>sha256sum</td>
<td>233</td>
</tr>
<tr>
<td>sha384sum</td>
<td>234</td>
</tr>
<tr>
<td>sha512sum</td>
<td>234</td>
</tr>
<tr>
<td>shuf</td>
<td>234</td>
</tr>
<tr>
<td>shutdown</td>
<td>235</td>
</tr>
<tr>
<td>sort</td>
<td>235</td>
</tr>
<tr>
<td>ss</td>
<td>236</td>
</tr>
<tr>
<td>strace</td>
<td>236</td>
</tr>
<tr>
<td>sysctl</td>
<td>237</td>
</tr>
<tr>
<td>systemctl</td>
<td>237</td>
</tr>
<tr>
<td>systemd-analyze</td>
<td>238</td>
</tr>
<tr>
<td>tcpflow</td>
<td>238</td>
</tr>
<tr>
<td>timedatectl</td>
<td>238</td>
</tr>
<tr>
<td>top</td>
<td>239</td>
</tr>
<tr>
<td>tree</td>
<td>239</td>
</tr>
<tr>
<td>ufw</td>
<td>239</td>
</tr>
<tr>
<td>ulimit</td>
<td>240</td>
</tr>
<tr>
<td>umask</td>
<td>240</td>
</tr>
<tr>
<td>uname</td>
<td>241</td>
</tr>
<tr>
<td>unexpand</td>
<td>241</td>
</tr>
<tr>
<td>useradd</td>
<td>242</td>
</tr>
<tr>
<td>userdel</td>
<td>242</td>
</tr>
<tr>
<td>usermod</td>
<td>243</td>
</tr>
<tr>
<td>vgcreate</td>
<td>243</td>
</tr>
<tr>
<td>wall</td>
<td>243</td>
</tr>
<tr>
<td>watch</td>
<td>244</td>
</tr>
<tr>
<td>whatis</td>
<td>244</td>
</tr>
<tr>
<td>wpa_cli</td>
<td>244</td>
</tr>
<tr>
<td>x11vnc</td>
<td>245</td>
</tr>
<tr>
<td>xclip</td>
<td>246</td>
</tr>
<tr>
<td>xdotool</td>
<td>246</td>
</tr>
<tr>
<td>xeyes</td>
<td>246</td>
</tr>
<tr>
<td>xinput</td>
<td>247</td>
</tr>
<tr>
<td>xrandr</td>
<td>247</td>
</tr>
<tr>
<td>xsel</td>
<td>248</td>
</tr>
<tr>
<td>xsetwacom</td>
<td>248</td>
</tr>
<tr>
<td>yaourt</td>
<td>249</td>
</tr>
</tbody>
</table>
yum ........................................................................... 249
zypper ........................................................................ 250

3 OSX ........................................................................... 251
airport ........................................................................ 251
archey ........................................................................ 251
base64 ........................................................................ 252
brew ........................................................................... 252
caffeinate ................................................................. 253
carthage ................................................................. 253
date ........................................................................... 253
dd ............................................................................... 254
defaults ................................................................. 254
diskutil ................................................................. 255
ditto ........................................................................ 255
drutil ................................................................. 255
du ........................................................................... 256
head ........................................................................ 256
hostname .............................................................. 257
launchctl .............................................................. 257
locate ........................................................................ 258
logger ........................................................................ 258
look ........................................................................ 259
md5 ........................................................................ 259
mdfind ................................................................. 259
netstat ................................................................. 260
networksetup ..................................................... 260
nm ........................................................................... 261
open ........................................................................ 261
pbcopy ..................................................................... 261
pbpaste ............................................................. 262
pmset ................................................................. 262
pod ........................................................................ 263
qlmanage ............................................................. 263
route ........................................................................ 263
say ........................................................................ 264
sed ........................................................................ 264
shutdown ............................................................. 265
sw_vers ............................................................... 266
sysctl ................................................................. 266
system_profiler ................................................. 266
systemsetup ....................................................... 267
top ........................................................................ 267
tree ........................................................................ 268
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>uname</td>
<td>268</td>
</tr>
<tr>
<td>w</td>
<td>269</td>
</tr>
<tr>
<td>wacaw</td>
<td>269</td>
</tr>
<tr>
<td>xattr</td>
<td>270</td>
</tr>
<tr>
<td>xcodebuild</td>
<td>270</td>
</tr>
<tr>
<td>xctool</td>
<td>271</td>
</tr>
<tr>
<td>xed</td>
<td>271</td>
</tr>
<tr>
<td>xsltproc</td>
<td>271</td>
</tr>
<tr>
<td>4 SUNOS</td>
<td>272</td>
</tr>
<tr>
<td>devfsadm</td>
<td>272</td>
</tr>
<tr>
<td>prctl</td>
<td>272</td>
</tr>
<tr>
<td>prstat</td>
<td>273</td>
</tr>
<tr>
<td>svcadm</td>
<td>273</td>
</tr>
<tr>
<td>svccfg</td>
<td>274</td>
</tr>
<tr>
<td>svcs</td>
<td>274</td>
</tr>
</tbody>
</table>
1 COMMON

7z

A file archiver with high compression ratio.

- Archive a file or folder:
  
  ```
  7z a archived.7z path/to/file
  ```

- Extract an existing 7z file with original directory structure:
  
  ```
  7z x archived
  ```

- Archive using a specific archive type:
  
  ```
  7z a -tzip|gzip|bzip2|tar|... archived path/to/file
  ```

- List available archive types:
  
  ```
  7z i
  ```

- List the contents of an archive file:
  
  ```
  7z l archived
  ```

7za

A file archiver with high compression ratio. A standalone version of 7z with support for fewer archive types.

- Archive a file or folder:
  
  ```
  7za a archived.7z path/to/file
  ```

- Extract an existing 7z file with original directory structure:
  
  ```
  7za x archived
  ```
- Archive using a specific archive type:

  7za a -t zip|gzip|bzip2|tar|... archived path/to/file

- List available archive types:

  7za i

- List the contents of an archive file:

  7za l archived

### 7zr

A file archiver with high compression ratio. A standalone version of 7z that only supports .7z files.

- Archive a file or folder:

  7zr a archived.7z path/to/file

- Extract an existing 7z file with original directory structure:

  7zr x archived.7z

- List the contents of an archive file:

  7zr l archived.7z

### ab

Apache Benchmarking tool. The simplest tool to perform a load testing.

- Execute 100 HTTP GET requests to given URL:

  ab -n 100 url

- Execute 100 HTTP GET requests, processing up to 10 requests concurrently, to given URL:

  ab -n 100 -c 10 url
**ack**

A search tool like grep, optimized for programmers.

- Find files containing “foo”:
  ```
  ack foo
  ```
- Find files in a specific language:
  ```
  ack --ruby each_with_object
  ```
- Count the total number of matches for the term “foo”:
  ```
  ack -ch foo
  ```
- Show the file names containing “foo” and number of matches in each file:
  ```
  ack -cl foo
  ```

**adb**

Android Debug Bridge: communicate with an Android emulator instance or connected Android devices.

- Check whether the adb server process is running and start it:
  ```
  adb start-server
  ```
- Terminate the adb server process:
  ```
  adb kill-server
  ```
- Start a remote shell in the target emulator/device instance:
  ```
  adb shell
  ```
- Push an Android application to an emulator/device:
  ```
  adb install -r path/to/file.apk
  ```
- Copy a file/folder from the target device:
  ```
  adb pull path/to/device_file_or_folder path/to/local_destination_folder
  ```
- Copy a file/folder to the target device:
  ```
  adb push path/to/local_file_or_folder path/to/device_destination_folder
  ```
- Get a list of connected devices:
  ```
  adb devices
  ```
**ag**

The Silver Searcher. Like ack, but faster.

- Find files containing "foo", and print the line matches in context:
  
  \texttt{ag \ foo}

- Find files containing “foo”, but only list the filenames:
  
  \texttt{ag -l \ foo}

- Find files containing “FOO” case-insensitively, and print only the match, rather than the whole line:

  \texttt{ag -i -o FOO}

- Find “foo” in files with a name matching “bar”:

  \texttt{ag \ foo -G bar}

- Find files whose contents match a regular expression:

  \texttt{ag '\^ba(r|z)$'}

- Find files with a name matching “foo”:

  \texttt{ag -g \ foo}

**alias**

Creates aliases – words that are replaced by a command string. Aliases expire with the current shell session, unless they’re defined in the shell’s configuration file, e.g. ```~/.bashrc```.

- Create a generic alias:

  \texttt{alias \ word="command"}

- View the command associated to a given alias:

  \texttt{alias \ word}

- Remove an aliased command:

  \texttt{unalias \ word}

- List all aliased words:

  \texttt{alias \ -p}

- Turn rm into an interactive command:

  \texttt{alias \ rm="rm \ -i"}

- Create la as a shortcut for ls -a:

  \texttt{alias \ la="ls \ -a"}
**ansible-playbook**

Execute tasks defined in playbook on remote machines over SSH.

- Run tasks in playbook:
  
  `ansible-playbook playbook`

- Run tasks in playbook with custom host inventory:
  
  `ansible-playbook playbook -i inventory_file`

**ansible**

Manage groups of computers remotely over SSH. Use the /etc/ansible/hosts file to add new groups/hosts.

- List hosts belonging to a group:
  
  `ansible group --list-hosts`

- Ping a group of hosts by invoking the ping module:
  
  `ansible group -m ping`

- Display facts about a group of hosts by invoking the setup module:
  
  `ansible group -m setup`

- Execute a command on a group of hosts by invoking command module with arguments:
  
  `ansible group -m command -a 'my command'`

- Execute a command with administrative privileges:
  
  `ansible group --become --ask-become-pass -m command -a 'my command'`

- Execute a command using a custom inventory file:
  
  `ansible group -i inventory_file -m command -a 'my command'`
**apg**

Creates arbitrarily complex random passwords.

- Create random passwords (default password length is 8):
  
apg

- Create a password with at least 1 symbol (S), 1 number (N), 1 uppercase (C), 1 lowercase (L):
  
apg -M SNCL

- Create a password with 16 characters:
  
apg -m 16

- Create a password with maximum length of 16:
  
apg -x 16

- Create a password that doesn’t appear in a dictionary (the dictionary file has to be provided):
  
apg -r dictionary_file

**apm**

Atom editor Package Manager. See atom.

- Install packages from http://atom.io/packages and themes from http://atom.io/themes:
  
apm install package_name

- Remove packages/themes:
  
apm remove package_name

- Upgrade packages/themes:
  
apm upgrade package_name
apropos

Search in manpages, for example to find a new command.

- Search for keyword:
  
apropos regular_expression

- Search without restricting output to terminal width:
  
apropos -l regular_expression

ar

Create, modify, and extract from archives (.a, .so, .o).

- Extract all members from an archive:
  
ar -x libfoo.a

- List the members of an archive:
  
ar -t libfoo.a

- Replace or add files to an archive:
  
ar -r libfoo.a foo.o bar.o baz.o

- Insert an object file index (equivalent to using ranlib):
  
ar -s libfoo.a

- Create an archive with files and an accompanying object file index:
  
ar -rs libfoo.a foo.o bar.o baz.o
**aria2c**

Fast download utility. Supports HTTP(S), FTP, SFTP, BitTorrent, and Metalink.

- Download a URI to a file:
  ```
  aria2c url
  ```

- Download from multiple sources:
  ```
  aria2c url_1 url_2
  ```

- Download the URIs listed in a file:
  ```
  aria2c -i filename
  ```

- Download with multiple connections:
  ```
  aria2c -s connections_num url
  ```

- FTP download with username and password:
  ```
  aria2c --ftp-user=username --ftp-passwd=password url
  ```

**arp**

Show and manipulate your system’s ARP cache.

- Show current arp table:
  ```
  arp -a
  ```

- Clear the entire cache:
  ```
  sudo arp -a -d
  ```

- Delete a specific entry:
  ```
  arp -d address
  ```

- Create an entry:
  ```
  arp -s address mac address
  ```
**assimp**

Command-line client for the Open Asset Import Library. Supports loading of 40+ 3D file formats, and exporting to several popular 3D formats.

- List all supported import formats:
  
  ```sh
  assimp listext
  ```

- List all supported export formats:
  
  ```sh
  assimp listexport
  ```

- Convert a file to one of the supported output formats, using the default parameters:
  
  ```sh
  assimp export input_file.stl output_file.obj
  ```

- Convert a file using custom parameters (the dox_cmd.h file in assimp’s source code lists available parameters):
  
  ```sh
  assimp export input_file.stl output_file.obj parameters
  ```

- Display a summary of a 3D file’s contents:
  
  ```sh
  assimp info path/to/file
  ```

- List all supported subcommands (“verbs”):
  
  ```sh
  assimp help
  ```

- Get help on a specific subcommand (e.g. the parameters specific to it):
  
  ```sh
  assimp subcommand --help
  ```

**atom**

A cross-platform pluggable text editor. Plugins are managed by `apm`.

- Open a file or folder:
  
  ```sh
  atom path/to/file_or_folder
  ```

- Open a file or folder in a new window:
  
  ```sh
  atom -n path/to/file_or_folder
  ```
**autojump**

Quickly jump among the directories you visit the most. Aliases like `j` or `jc` are provided for even less typing.

- Jump to a directory that contains the given pattern:
  
  `j pattern`

- Jump to a sub-directory (child) of the current directory that contains the given pattern:
  
  `jc pattern`

- Remove non-existing directories from the autojump database:
  
  `j --purge`

- Show the entries in the autojump database:
  
  `j -s`

**autossh**

Runs, monitors and restarts SSH connections. Auto-reconnects to keep port forwarding tunnels up. Accepts all ssh flags.

- Open an SSH session, restarting when a monitoring port fails return data:
  
  `autossh -M monitor_port ssh_command`

- Open an SSH session which forwards a local port to a remote one, restarting if necessary:
  
  `autossh -M monitor_port -L local_port:localhost:remote_port user@host`

- Fork before executing ssh (runs in the background) and don’t open a remote shell:
  
  `autossh -f -M monitor_port -N ssh_command`

- Run autossh in the background, with no monitoring port, instead relying on SSH keep-alives every 10 seconds to detect failure:
  
  `autossh -f -M 0 -N -o "ServerAliveInterval 10" -o "ServerAliveCountMax 3" ssh_command`

- Run autossh in the background, with no monitoring port, no remote shell, exiting if the port forward fails:
  
  `autossh -f -M 0 -N -o "ServerAliveInterval 10" -o "ServerAliveCountMax 3" -o ExitOnForwardFailure=yes -L local_port:localhost:remote_port user@host`

- Run autossh in the background with debug output logged to a file and ssh verbose output logged to a second file:
  
  `AUTOSSH_DEBUG=1 AUTOSSH_LOGFILE=log_file autossh -f -M monitor_port -v -E ssh_log_file ssh_command`
**avrdude**

Driver program for Atmel AVR microcontrollers programming.

- Read AVR microcontroller:
  
  ```
  avrdude -p AVR_device -c programmer -U flash:r:file.hex:i
  ```

- Write AVR microcontroller:
  
  ```
  avrdude -p AVR_device -c programmer -U flash:w:file.hex
  ```

- List available AVR devices:
  
  ```
  avrdude -p ?
  ```

- List available AVR programmers:
  
  ```
  avrdude -c ?
  ```

**awk**

A versatile programming language for working on files.

- Print the fifth column (a.k.a. field) in a space-separated file:
  
  ```
  awk '{print $5}' filename
  ```

- Print the second column of the lines containing "something" in a space-separated file:
  
  ```
  awk '/something/ {print $2}' filename
  ```

- Print the last column of each line in a file, using a comma (instead of space) as a field separator:
  
  ```
  awk -F ',' '{print $NF}' filename
  ```

- Sum the values in the first column of a file and print the total:
  
  ```
  awk '{s+=$1} END {print s}' filename
  ```

- Sum the values in the first column and pretty-print the values and then the total:
  
  ```
  awk '{s+=$1; print $1} END {print "--------"; print s}' filename
  ```
**aws s3**

CLI for AWS S3 - provides storage through web services interfaces.

- Show files in a bucket:
  
  ```
  aws s3 ls bucket_name
  ```

- Sync files and folders from local to bucket:
  
  ```
  aws s3 sync path/to/files s3://bucket_name
  ```

- Sync files and folders from bucket to local:
  
  ```
  aws s3 sync s3://bucket_name path/to/target
  ```

- Sync files and folders with exclusions:
  
  ```
  aws s3 sync path/to/files s3://bucket_name --exclude path/to/file --exclude path/to/folder/*
  ```

- Remove file from bucket:
  
  ```
  aws s3 rm s3://bucket/path/to/file
  ```

- Preview changes only:
  
  ```
  aws s3 any_command --dryrun
  ```

**axel**

Download accelerator. Supports HTTP, HTTPS, and FTP.

- Download a URL to a file:
  
  ```
  axel url
  ```

- Download and specify filename:
  
  ```
  axel url -o filename
  ```

- Download with multiple connections:
  
  ```
  axel -n connections_num url
  ```

- Search for mirrors:
  
  ```
  axel -S mirrors_num url
  ```

- Limit download speed (bytes per second):
  
  ```
  axel -s speed url
  ```
**base32**

Encode or decode file or standard input, to standard output.

- Encode a file:
  
  base32 filename

- Decode a file:
  
  base32 -d filename

- Encode from stdin:
  
  somecommand | base32

- Decode from stdin:
  
  somecommand | base32 -d

**base64**

Encode or decode file or standard input, to standard output.

- Encode a file:
  
  base64 filename

- Decode a file:
  
  base64 -d filename

- Encode from stdin:
  
  somecommand | base64

- Decode from stdin:
  
  somecommand | base64 -d

**basename**

Returns non-directory portion of a pathname.

- Show only the file name from a path:
  
  basename path/to/file

- Show only the file name from a path, with a suffix removed:
  
  basename path/to/file suffix
**bash**

Bourne-Again SHell. sh-compatible command line interpreter.

- Start interactive shell:
  
  bash

- Execute a command:

  bash -c "command"

- Run commands from a file:

  bash file.sh

- Run commands from STDIN:

  bash -s

- Print the version information of bash (use echo $BASH_VERSION to show just the version string):

  bash --version

---

**bashmarks**

Save and jump to commonly used directories using 1 character commands.

- List available bookmarks:

  1

- Save the current folder as “bookmark_name”:

  s bookmark_name

- Go to a bookmarked folder:

  g bookmark_name

- Print a bookmarked folder’s contents:

  p bookmark_name

- Delete a bookmark:

  d bookmark_name
**bc**

Calculator.
- Run calculator in interactive mode using the standard math library:
  
  `bc -l`

- Calculate the result of an expression:
  
  `bc <<< "(1 + 2) * 2 ^ 2"`

- Calculate expression and force number of decimal places to 10:
  
  `bc <<< "scale=10; 5 / 3"`

- Calculate expression with sine and cosine using mathlib:
  
  `bc -l <<< "s(1) + c(1)"

**bedtools**

A swiss-army knife of tools for genomic-analysis tasks. Used to intersect, group, convert and count data in BAM, BED, GFF/GTF, VCF format.

- Intersect two files with respect to the sequences’ strand and save the result to \{{{path/to/output_file}}}:
  
  `bedtools intersect -a path/to/file_1 -b path/to/file_2 -s > path/to/output_file`

- Intersect two files with a left outer join, i.e. report each feature from \{{{file_1}}} and NULL if no overlap with \{{{file_2}}}:
  
  `bedtools intersect -a path/to/file_1 -b path/to/file_2 -lof > path/to/output_file`

- Using more efficient algorithm to intersect two pre-sorted files:
  
  `bedtools intersect -a path/to/file_1 -b path/to/file_2 -sorted > path/to/output_file`

- Group file \{{{path/to/file}}} based on the first three and the fifth column and summarize the sixth column by summing it up:
  
  `bedtools groupby -i path/to/file -c 1-3,5 -g 6 -o sum`

- Convert bam-formated file to a bed-formated one:
  
  `bedtools bamtobed -i path/to/file.bam > path/to/file.bed`

- Find for all features in \{{{file_1}}}\.bed the closest one in \{{{file_2}}}\.bed and write their distance in an extra column (input files must be sorted):
  
  `bedtools closest -a path/to/file_1.bed -b path/to/file_2.bed -d`
**bg**

Resumes suspended jobs and keeps them running in the background.

- Resume most recently suspended background job running in the background:
  
  `bg`

- Resume a specific job running in the background:
  
  `bg job_id`

**blender**

Command-line interface to the Blender 3D computer graphics application. Arguments are executed in the order they are given.

- Render all frames of an animation in the background, without loading the UI (output is saved to /tmp):

  `blender -b filename.blend -a`

- Render an animation using a specific image naming pattern, in a path relative (//) to the .blend file:

  `blender -b filename.blend -o //render/frame_###.png -a`

- Render the 10th frame of an animation as a single image, saved to an existing folder (absolute path):

  `blender -b filename.blend -o /path/to/output_folder -f 10`

- Render the second last frame in an animation as a JPEG image, saved to an existing folder (relative path):

  `blender -b filename.blend -o //output_folder -F JPEG -f -2`

- Render the animation of a specific scene, starting at frame 10 and ending at frame 500:

  `blender -b filename.blend -S scene_name -s 10 -e 500 -a`

- Render an animation at a specific resolution, by passing a Python expression:

  `blender -b filename.blend --python-expr 'import bpy; bpy.data.scenes[0].render.resolution_percentage = 25' -a`

- Start an interactive Blender session in the terminal with a python console (do import bpy after starting):

  `blender -b --python-console`
**bmaptool**

Create or Copy blockmaps intelligently (and therefore faster than cp or dd).

- Create a blockmap from image file:
  
  `bmaptool create -o blockmap.bmap source.img`

- Copy an image file into sdb:
  
  `bmaptool copy --bmap blockmap.bmap source.img /dev/sdb`

- Copy a compressed image file into sdb:
  
  `bmaptool copy --bmap blockmap.bmap source.img.gz /dev/sdb`

- Copy an image file into sdb without using a blockmap:
  
  `bmaptool copy --nobmap source.img /dev/sdb`

**bower**

A package manager optimized for front-end web development. A package can be a GitHub user/repo shorthand, a Git endpoint, a URL or a registered package.

- Install a project’s dependencies, listed in its bower.json:
  
  `bower install`

- Install one or more packages to the bower_components directory:
  
  `bower install package package`

- Uninstall packages locally from the bower_components directory:
  
  `bower uninstall package package`

- List local packages and possible updates:
  
  `bower list`

- Display help information about a bower command:
  
  `bower help command`

- Create a bower.json file for your package:
  
  `bower init`

- Install a specific dependency version, and add it to bower.json:
  
  `bower install local_name=package#version --save`
bundle

Dependency manager for the Ruby programming language.

- Install all gems defined in the gemfile expected in the working directory:
  
  `bundle install`

- Update all gems by the rules defined in the gemfile and regenerate gemfile.lock:
  
  `bundle update`

- Update one specific gem defined in the gemfile:
  
  `bundle update --source gemname`

- Create a new gem skeleton:
  
  `bundle gem gemname`

c99

Compiles C programs according to the ISO C standard.

- Compile source file(s) and create an executable:
  
  `c99 file.c`

- Compile source file(s) and create an executable with a custom name:
  
  `c99 -o executable_name file.c`

- Compile source file(s) and create object file(s):
  
  `c99 -c file.c`

- Compile source file(s), link with object file(s), and create an executable:
  
  `c99 file.c file.o`
**cabal**

Command line interface to the Haskell package infrastructure (Cabal). Manage Haskell projects and Cabal packages from the Hackage package repository.

- Search and list packages from Hackage:
  
cabal list search_string

- Show information about a package:
  
cabal info package_name

- Download and install a package:
  
cabal install package_name

- Create a new Haskell project in the current directory:
  
cabal init

- Build the project in the current directory:
  
cabal build

- Run tests of the project in the current directory:
  
cabal test

**cal**

Prints calendar information.

- Display a calendar for the current month:
  
cal

- Display a calendar for a specific month:
  
cal -m month_number

- Display a 12 month calendar for the current year:
  
cal -y

- Display a 12 month calendar for a specific year:
  
cal 2016

- Display date of Easter (western churches):
  
cal -e year
**calibre-server**

A server application that can be used to distribute ebooks over a network. Ebooks must be imported into the library using the GUI or calibredb before. Part of the Calibre ebook library.

- Start a server to distribute ebooks. Access at http://localhost:8080:
  
  `calibre-server`

- Start server on different port. Access at http://localhost:port:
  
  `calibre-server --port` `port`

- Password protect the server:

  `calibre-server --username` `username` `--password` `password`

**calibredb**

Tool to manipulate the your ebook database. Part of the Calibre ebook library.

- List ebooks in the library with additional information:

  `calibredb list`

- Search for ebooks displaying additional information:

  `calibredb list --search` `search_term`

- Search for just ids of ebooks:

  `calibredb search` `search_term`

- Add one or more ebooks to the library:

  `calibredb add` `file1` `file2` ...

- Remove one or more ebooks from the library. You need ebook-ids (see above):

  `calibredb remove` `id1` `id2` ...
**cargo**

Rust package manager. Manage Rust projects and their module dependencies (crates).

- Search for crates:
  
  `cargo search search_string`

- Install a crate:
  
  `cargo install crate_name`

- List installed crates:
  
  `cargo install --list`

- Create a new binary Rust project in the current directory:
  
  `cargo init --bin`

- Create a new library Rust project in the current directory:
  
  `cargo init`

- Build the Rust project in the current directory:
  
  `cargo build`

- Build with multiple parallel jobs:
  
  `cargo build -j jobs`

**cat**

Print and concatenate files.

- Print the contents of a file to the standard output:
  
  `cat file`

- Concatenate several files into the target file:
  
  `cat file1 file2 > target_file`

- Append several files into the target file:
  
  `cat file1 file2 >> target_file`

- Number all output lines:
  
  `cat -n file`
**cd**

Change the current working directory.

- Go to the given directory:
  
  ```
  cd path/to/directory
  ```

- Go to home directory of current user:
  
  ```
  cd
  ```

- Go up to the parent of the current directory:
  
  ```
  cd ..
  ```

- Go to the previously chosen directory:
  
  ```
  cd -
  ```

**chgrp**

Change group ownership of files and folders.

- Change the owner of a file/folder:
  
  ```
  chgrp group path/to/file
  ```

- Recursively change the owner of a folder and its contents:
  
  ```
  chgrp -R group path/to/folder
  ```

- Change the owner of a symbolic link:
  
  ```
  chgrp -h user path/to/symlink
  ```

- Change the owner of a file/folder to match a reference file:
  
  ```
  chgrp --reference=path/to/reference_file path/to/file
  ```
**chmod**

Change the access permissions of a file or directory.

- Give the [u]ser who owns a file the right to e[x]ecute it:
  
  `chmod u+x file`

- Give the user rights to [r]ead and [w]rite to a file/directory:
  
  `chmod u+rw file`

- Remove executable rights from the [g]roup:
  
  `chmod g-x file`

- Give [a]ll users rights to read and execute:
  
  `chmod a+rx file`

- Give [o]thers (not in the file owner’s group) the same rights as the group:
  
  `chmod o=g file`

**chown**

Change user and group ownership of files and folders.

- Change the owner user of a file/folder:
  
  `chown user path/to/file`

- Change the owner user and group of a file/folder:
  
  `chown user:group path/to/file`

- Recursively change the owner of a folder and its contents:
  
  `chown -R user path/to/folder`

- Change the owner of a symbolic link:
  
  `chown -h user path/to/symlink`

- Change the owner of a file/folder to match a reference file:
  
  `chown --reference=path/to/reference_file path/to/file`
**chsh**

Change user’s login shell.
- Change shell:
  
  `chsh -s path/to/shell_binary username`

**cksum**

Calculates CRC checksums and byte counts of a file. Note, on old UNIX systems the CRC implementation may differ.
- Display a 32 bit checksum, size in bytes and filename:
  
  `cksum filename`

**clang**

Compiler for C, C++, and Objective-C source files. Can be used as a drop-in replacement for GCC.
- Compile a source code file into an executable binary:
  
  `clang input_source.c -o output_executable`
- Activate output of all errors and warnings:
  
  `clang input_source.c -Wall -o output_executable`
- Include libraries located at a different path than the source file:
  
  `clang input_source.c -o output_executable -Iheader_path -Llibrary_path -llibrary_name`

**cloc**

Count, and compute differences of, lines of source code and comments.
- Count all the lines of code in a directory:
  
  `cloc /path/to/directory`
- Count all the lines of code in a directory, displaying a progress bar during the counting process:
  
  `cloc --progress=1 /path/to/directory`
- Compare 2 directory sturctures and count the differences between them:
  
  `cloc --diff /directory/one /directory/two`
**cmp**

Compare two files.

- Find the byte number and line number of the first difference between the files:
  
  ```bash
cmp file1 file2
  ```

- Find the byte number and differing bytes of every difference:
  
  ```bash
cmp -l file1 file2
  ```

**column**

Format standard input or file into multiple columns. Rows are filled before columns; default separator is whitespace.

- Format output for a 30 characters wide display:
  
  ```bash
  printf "header1 header2\nbar foo\n" | column -c 30
  ```

- Specify column delimiter character for the -t option (i.e. "," for csv); default is whitespace:
  
  ```bash
  printf "header1,header2\nbar,foo\n" | column -s ,
  ```

- Split columns automatically and auto-align in a tabular format:
  
  ```bash
  printf "header1 header2\nbar foo\n" | column -t
  ```

- Fill columns before filling rows:
  
  ```bash
  printf "header1\nbar\nfoobar\n" | column -c 30 -x
  ```

**comm**

Select or reject lines common to two files. Both files must be sorted.

- Produce three tab-separated columns: lines only in first file, lines only in second file and common lines:
  
  ```bash
  comm file1 file2
  ```

- Print only lines common to both files:
  
  ```bash
  comm -12 file1 file2
  ```
- Print only lines common to both files, reading one file from stdin:
  ```
cat file1 | comm -12 - file2
  ```
- Get lines only found in first file, saving the result to a third file:
  ```
comm -23 file1 file2 > file1_only
  ```
- Print lines only found in second file, when the files aren’t sorted:
  ```
comm -13 <(sort file1) <(sort file2)
  ```

**conda**

Package, dependency and environment management for any programming language.

- Create a new environment, installing named packages into it:
  ```
conda create --name environment_name python=2.7 matplotlib
  ```
- List all environments:
  ```
conda info --envs
  ```
- Load or unload an environment:
  ```
source activate|deactivate environment_name
  ```
- Delete an environment (remove all packages):
  ```
conda remove --name environment_name --all
  ```
- Search conda channels for a package by name:
  ```
conda search package_name
  ```
- Install packages into the current environment:
  ```
conda install python=3.4 numpy
  ```
- List currently installed packages in current environment:
  ```
conda list
  ```
- Delete unused packages and caches:
  ```
conda clean --all
  ```
**consul-kv**

Distributed key-value store with health checking and service discovery.

- Read a value from the key-value store:
  ```
  consul kv get key
  ```
- Store a new key-value pair:
  ```
  consul kv put key value
  ```
- Delete a key-value pair:
  ```
  consul kv delete key
  ```

**consul**

Distributed key-value store with health checking and service discovery.

- Check the Consul version:
  ```
  consul --version
  ```
- Show general help:
  ```
  consul --help
  ```
- Show help for a sub-command:
  ```
  consul sub-command --help
  ```

**convert**

Imagemagick image conversion tool.

- Convert an image from JPG to PNG:
  ```
  convert image.jpg image.png
  ```
- Scale an image 50% it's original size:
  ```
  convert image.png -resize 50% image2.png
  ```
- Scale an image keeping the original aspect ratio to a maximum dimension of 640x480:
  ```
  convert image.png -resize 640x480 image2.png
  ```
- Horizontally append images:
  ```
  convert image1.png image2.png image3.png +append image123.png
  ```
**convmv**

Convert filenames (NOT file content) from one encoding to another.

- Test filename encoding conversion (don’t actually change the filename):
  
  `convmv -f from_encoding -t to_encoding input_file`

- Convert filename encoding and rename the file to the new encoding:
  
  `convmv -f from_encoding -t to_encoding --notest input_file`

**cordova**

Mobile apps with HTML, CSS & JS.

- Create a cordova project:
  
  `cordova create path package.name project.name`

- Display the current workspace status:
  
  `cordova info`

- Add a cordova platform:
  
  `cordova platform add platform`

- Remove a cordova platform:
  
  `cordova platform remove platform`

- Add a cordova plugin:
  
  `cordova plugin add pluginid`

- Remove a cordova plugin:
  
  `cordova plugin remove pluginid`
cowsay

Generate an ASCII character like a cow or sheep saying or thinking something. Available characters are stored in the /usr/share/cowsay on Linux. And /usr/local/share/cows/ on Mac.

- Print an ASCII cow saying "Hello world!":
  ```
cowsay "Hello world!"
  ```
- Print an ASCII dragon saying "Hello!":
  ```
  echo "Hello!" | cowsay -f dragon
  ```
- Print a stoned thinking ASCII cow:
  ```
cowthink -s "I'm just a cow, not a great thinker ..."
  ```
- Print out a list of all characters with cowsay:
  ```
  ls -1 cowsay_character_directory | rev | cut -c 5- | rev | xargs -I _ cowsay -f _
  ```

cp

Copy files.

- Copy files in arbitrary locations:
  ```
cp /path/to/original /path/to/copy
  ```
- Copy a file to a parent directory:
  ```
cp /path/to/original ..//path/to/copy
  ```
- Copy directories recursive using the option -r:
  ```
cp -r /path/to/original /path/to/copy
  ```
- Show files as they are copied:
  ```
cp -vr /path/to/original /path/to/copy
  ```
- Make a copy of a file, adding an extension:
  ```
cp file.html{,.backup}
  ```
- Make a copy of a file, changing the extension:
  ```
cp file.{html,backup}
  ```
**cpio**

Copies files in and out of archives. Supports the following archive formats: cpio’s custom binary, old ASCII, new ASCII, crc, HPUX binary, HPUX old ASCII, old tar, and POSIX.1 tar.

- Take a list of file names from standard input and add them [o]nto an archive in cpio’s binary format:
  ```
  echo "file1 file2 file3" | cpio -o > archive.cpio
  ```
- Copy all files and folders in a directory and add them [o]nto an archive, in [v]erbose mode:
  ```
  find path/to/directory | cpio -ov > archive.cpio
  ```
- P[li]ck all files from an archive, generating [d]irectories where needed, in [v]erbose mode:
  ```
  cpio -idv < archive.cpio
  ```

**cppcheck**

A static analysis tool for C/C++ code. Instead of syntax errors, it focuses on the types of bugs that compilers normally do not detect.

- Recursively check the current folder, showing progress on the screen and logging error messages to a file:
  ```
  cppcheck . 2> cppcheck.log
  ```
- Recursively check a given folder, and don’t print progress messages:
  ```
  cppcheck --quiet path/to/folder
  ```
- Check a given file, specifying which tests to perform (by default only errors are shown):
  ```
  cppcheck --enable=error|warning|style|performance|portability|information|all path/to/file.cpp
  ```
- List available tests, filtered by a given search pattern:
  ```
  cppcheck --errorlist | grep "search pattern"
  ```
- Check a given file, ignoring specific tests:
  ```
  cppcheck --suppress=test_id1 --suppress=test_id2 path/to/file.cpp
  ```
- Check the current folder, providing paths for include files located outside it (e.g. external libraries):
  ```
  cppcheck -I include/folder_1 -I include/folder_2 .
  ```
- Check a Microsoft Visual Studio project (*.vcxproj) or solution (*.sln):
  ```
  cppcheck --project=path/to/project.sln
  ```
**crontab**

Schedule cron jobs to run on a time interval for the current user. Job definition format: “(min) (hour) (day_of_month) (month) (day_of_week) command_to_execute”.

- Edit the crontab file for the current user:
  ```
  crontab -e
  ```

- View a list of existing cron jobs for current user:
  ```
  crontab -l
  ```

- Remove all cron jobs for the current user:
  ```
  crontab -r
  ```

- Sample job which runs at 10:00 every day. * means any value:
  ```
  0 10 * * * path/to/script.sh
  ```

- Sample job which runs every minute on the 3rd of April:
  ```
  * * 3 Apr * path/to/script.sh
  ```

- Sample job which runs at 02:30 every Friday:
  ```
  30 2 * * Fri path/to/script.sh
  ```

**csvclean**

Finds and cleans common syntax errors in CSV files. Included in csvkit.

- Clean a CSV file:
  ```
  csvclean bad.csv
  ```

- List locations of syntax errors in a CSV file:
  ```
  csvclean -n bad.csv
  ```
**csvcut**

Filter and truncate CSV files. Like Unix’s cut command, but for tabular data. Included in csvkit.

- Print indices and names of all columns:

  \[ \texttt{csvcut -n data.csv} \]

- Extract the first and third columns:

  \[ \texttt{csvcut -c 1,3 data.csv} \]

- Extract all columns except the fourth one:

  \[ \texttt{csvcut -C 4 data.csv} \]

- Extract the columns named “id” and “first name” (in that order):

  \[ \texttt{csvcut -c id,"first name" data.csv} \]

**csvformat**

Convert a CSV file to a custom output format. Included in csvkit.

- Convert to a tab-delimited file (TSV):

  \[ \texttt{csvformat -T data.csv} \]

- Convert delimiters to a custom character:

  \[ \texttt{csvformat -D \"custom_character\" data.csv} \]

- Convert line endings to carriage return (^M) + line feed:

  \[ \texttt{csvformat -M \"\r\n\" data.csv} \]

- Minimize use of quote characters:

  \[ \texttt{csvformat -U 0 data.csv} \]

- Maximize use of quote characters:

  \[ \texttt{csvformat -U 1 data.csv} \]
**csvgrep**

Filter CSV rows with string and pattern matching. Included in csvkit.

- Find rows that have a certain string in column 1:
  
  `csvgrep -c 1 -m string_to_match data.csv`

- Find rows in which columns 3 or 4 match a certain regex pattern:
  
  `csvgrep -c 3,4 -r regex_pattern data.csv`

- Find rows in which the “name“ column does NOT include the string ”John Doe“:
  
  `csvgrep -i -c name -m "John Doe" data.csv`

**csvlook**

Render a CSV file in the console as a fixed-width table. Included in csvkit.

- View a CSV file:
  
  `csvlook data.csv`

**csvpy**

Loads a CSV file into a Python shell. Included in csvkit.

- Load a CSV file into a CSVKitReader object:
  
  `csvpy data.csv`

- Load a CSV file into a CSVKitDictReader object:

  `csvpy --dict data.csv`
**csvsort**

Sorts CSV files. Included in csvkit.

- Sort a CSV file by column 9:
  
  `csvsort -c 9 data.csv`

- Sort a CSV file by the “name” column in descending order:
  
  `csvsort -r -c name data.csv`

- Sort a CSV file by column 2, then by column 4:
  
  `csvsort -c 2,4 data.csv`

- Sort a CSV file without inferring data types:
  
  `csvsort --no-inference -c columns data.csv`

**csvstat**

Print descriptive statistics for all columns in a CSV file. Included in csvkit.

- Show all stats for all columns:
  
  `csvstat data.csv`

- Show all stats for columns 2 and 4:
  
  `csvstat -c 2,4 data.csv`

- Show sums for all columns:
  
  `csvstat --sum data.csv`

- Show the max value length for column 3:
  
  `csvstat -c 3 --len data.csv`

- Show the number of unique values in the “name” column:
  
  `csvstat -c name --unique data.csv`
**curl**

Transfers data from or to a server. Supports most protocols including HTTP, FTP, POP3.

- Download the contents of an URL to a file:
  
  ```
  curl http://example.com -o filename
  ```

- Download a file saving the output under the filename indicated by the URL:
  
  ```
  curl -O http://example.com/filename
  ```

- Download a file, following [L]ocation redirects, and automatically [C]ontinuing (resuming) a previous file transfer:
  
  ```
  curl -O -L -C - http://example.com/filename
  ```

- Send form-encoded data (POST request of type application/x-www-form-urlencoded):
  
  ```
  curl -d 'name=bob' http://example.com/form
  ```

- Send data, specifying a custom HTTP method, and including an extra header:
  
  ```
  curl -d '{"name":"bob"}' -X PUT -H 'Content-Type: application/json' http://example.com/users/1234
  ```

- Pass a user name and password for server authentication and show headers info only:
  
  ```
  curl -u myusername:mypassword -I http://example.com
  ```

- Pass client certificate and key for a secure resource:
  
  ```
  curl -v -key key.pem -cacert ca.pem -cert client.pem -k https://example.com
  ```

**cut**

Cut out fields from STDIN or files.

- Cut out the first sixteen characters of each line of STDIN:
  
  ```
  cut -c 1-16
  ```

- Cut out the first sixteen characters of each line of the given files:
  
  ```
  cut -c 1-16 file
  ```
- Cut out everything from the 3rd character to the end of each line:
  
  `cut -c3-`

- Cut out the fifth field of each line, using a colon as a field delimiter (default delimiter is tab):
  
  `cut -d':' -f5`

- Cut out the 2nd and 10th fields of each line, using a semicolon as a delimiter:
  
  `cut -d';' -f2,10`

- Cut out the fields 3 through 7 of each line, using a space as a delimiter:
  
  `cut -d' ' -f3-7`

**deluser**

Remove a user account or remove a user from a group.

- Remove a user:
  
  `deluser name`

- Remove a user along with their home directory and mail spool:
  
  `deluser -r name`

- Remove a user from a group:
  
  `deluser name group`

**df**

Gives an overview of the file system disk space usage.

- Display all file systems and their disk usage:
  
  `df`

- Display all file systems and their disk usage in human readable form:
  
  `df -h`
**dhcpwn**

Test DHCP IP exhaustion attacks and sniff local DHCP traffic.

- Flood the network with IP requests:
  
  ```bash
dhcpwn --interface network_interface flood --count number_of_requests
  ``

- Sniff local DHCP traffic:
  
  ```bash
dhcpwn --interface network_interface sniff
  ``

**diff**

Compare files and directories.

- Compare files:
  
  ```bash
diff file1 file2
  ``

- Compare files, ignoring white spaces:
  
  ```bash
diff -w file1 file2
  ``

- Compare files, showing differences side by side:
  
  ```bash
diff -y file1 file2
  ``

- Compare directories recursively:
  
  ```bash
diff -r directory1 directory2
  ``

- Compare directories, only showing the names of files that differ:
  
  ```bash
diff -rq directory1 directory2
  ```
**dig**

DNS Lookup utility.

- Lookup the IP(s) associated with a hostname (A records):
  
  ```
  dig +short hostname.com
  ```

- Lookup the mail server associated with a given domain name (MX record):
  
  ```
  dig +short hostname.com MX
  ```

- Specify an alternate DNS server to query (8.8.8.8 is google’s public DNS):
  
  ```
  dig @8.8.8.8 hostname.com
  ```

- Perform a reverse DNS lookup on an IP address (PTR record):
  
  ```
  dig -x 8.8.8.8
  ```

- Find authoritative name servers for the zone and display SOA records:
  
  ```
  dig +nssearch hostname.com
  ```

**dirs**

Displays or manipulatesthe directory stack. The directory stack is a list of recently visited directories that can be manipulated with the pushd and popd commands.

- Display the directory stack with a space between each entry:
  
  ```
  dirs
  ```

- Display the directory stack with one entry per line:
  
  ```
  dirs -p
  ```

- Display only the nth entry in the directory stack, starting at 0:
  
  ```
  dirs +N
  ```

- Clear the directory stack:
  
  ```
  dirs -c
  ```
**docker**

Manage Docker containers and images.

- List of running docker containers:
  
  ```
  docker ps
  ```

- List all docker containers (running and stopped):
  
  ```
  docker ps -a
  ```

- Start a container:
  
  ```
  docker start container
  ```

- Stop a container:
  
  ```
  docker stop container
  ```

- Start a container from an image and get a shell inside of it:
  
  ```
  docker run -it image bash
  ```

- Run a command inside of an already running container:
  
  ```
  docker exec container command
  ```

**dokku**

Docker powered mini-Heroku (PaaS). Easily deploy multiple apps to your server in different languages using a single git-push command.

- List running apps:
  
  ```
  dokku apps
  ```

- Create an app:
  
  ```
  dokku apps:create app_name
  ```

- Remove an app:
  
  ```
  dokku apps:destroy app_name
  ```

- Install plugin:
  
  ```
  dokku plugin:install full_repo_url
  ```

- Link database to an app:
  
  ```
  dokku db:link db_name app_name
  ```
drush

A command-line shell and scripting interface for Drupal.
- Download module "foo":
  drush dl foo
- Download version 7.x-2.1-beta1 of module "foo":
  drush dl foo-7.x-2.1-beta1
- Enable module "foo":
  drush en foo
- Disable module "foo":
  drush dis foo
- Clear all caches:
  drush cc all
- Clear CSS and JavaScript caches:
  drush cc css-js

ebook-convert

Can be used to convert ebooks between common formats, e.g., pdf, epub and mobi. Part of the Calibre ebook library tool.
- Convert an ebook into another format:
  ebook-convert source destination

echo

Print given arguments.
- Print a text message. Note: quotes are optional:
  echo "Hello World"
- Print a message with environment variables:
  echo "My path is $PATH"
- Print a message without the trailing newline:
  echo -n "Hello World"
- Enable interpretation of backslash escapes (special characters):
  echo -e "Column 1\tColumn 2"
**ed**

The original Unix text editor.

- Start ed, editing an empty document (which can be saved as a new file in the current directory):
  
ed

- Start ed, editing an empty document, with : as a command prompt indicator:
  
ed -p :

- Start ed editing an existing file (this shows the byte count of the loaded file):
  
ed -p : path/to/file

- Toggle the printing of error explanations. (By default, explanations are not printed and only a ? appears):
  
H

- Add text to the current document. Mark completion by entering a period by itself in a new line:
  
a<Enter>text_to_insert<Enter>.

- Print the entire document (, is a shortcut to the range 1, which covers the start to the end of the document):
  
, p

- Write the current document to a new file (the filename can be omitted if ed was called with an existing file):
  
w filename

- Quit ed:
  
q
**electrum**

Ergonomic Bitcoin wallet and private key management.

- Create a new wallet:
  
  `electrum -w new_wallet.dat create`

- Restore an existing wallet from seed offline:
  
  `electrum -w recovery_wallet.dat restore -o`

- Create a signed transaction offline:
  
  `electrum mktx recipient amount -f 0.0000001 -F from -o`

- Display all wallet receiving addresses:
  
  `electrum listaddresses -a`

- Sign a message:
  
  `electrum signmessage address message`

- Verify a message:
  
  `electrum verifymessage address signature message`

- Connect only to a specific electrum-server instance:
  
  `electrum -p socks5:127.0.0.1:9050 -s 56ckl5obj37gypcu.onion:50001:t -1`

**emacs**

The extensible, customizable, self-documenting, real-time display editor.

- Open emacs in console mode (without X window):
  
  `emacs -nw`

- Open a file in emacs:
  
  `emacs filename`

- Exit emacs:
  
  `C-x C-c`
**enca**

Detect and convert encoding of text files.

- Detect file(s) encoding according to your system’s locale:
  
  `enca file(s)`

- Detect file(s) encoding; -L option tells enca the current language; language is in the POSIX/C locale format, e.g. zh_CN, en_US etc:
  
  `enca -L language file(s)`

- Convert file(s) to specified encoding:
  
  `enca -L language -x to_encoding file(s)`

- Save original_file as new_file and convert new_file to specified encoding:
  
  `enca -L language -x to_encoding < original_file > new_file`

**env**

Show the environment or run a program in a modified environment.

- Show the environment:
  
  `env`

- Run a program. Often used in scripts after the shebang (#!) for looking up the path to the program:
  
  `env program`

- Clear the environment and run a program:
  
  `env -i program`

- Remove variable from the environment and run a program:
  
  `env -u variable program`

- Set a variable and run a program:
  
  `env variable=value program`
**espeak**

Uses text-to-speech to speak through the default sound device.

- Speak a phrase aloud:

  espeak "I like to ride my bike."

- Speak a file aloud:

  espeak -f filename

- Save output to a WAV audio file, rather than speaking it directly:

  espeak -w filename.wav "It's GNU plus Linux"

- Use a different voice:

  espeak -v voice

**exiftool**

Read and write meta information in files.

- Remove all EXIF metadata from the given files:

  exiftool -All= file

- Increase time photo taken by 1 hour in directory:

  exiftool "-AllDates+=0:0:0 1:0:0" directory

- Decrease time photo taken by 1 day and 2 hours on JPEGs only:

  exiftool "-AllDates-=0:0:1 2:0:0" -ext jpg

- Change only DateTimeOriginal by -1.5 hours & do not keep backups:

  exiftool -DateTimeOriginal-=1.5 -overwrite_original

- Rename all JPEGs according to a DateTimeOriginal recursively:

  exiftool '-filename<DateTimeOriginal' -d %Y-%m-%d_%H-%M-%S%%l%c.%e directory -r -ext jpg
**fdupes**

Finds duplicate files in a given set of directories.

- Search a single directory:
  
  `fdupes directory`

- Search multiple directories:
  
  `fdupes directory1 directory2`

- Search all directories recursively:
  
  `fdupes -r directory`

- Search multiple directories, one recursively:
  
  `fdupes directory1 -R directory2`

**ffmpeg**

Video conversion tool.

- Extract the sound from a video and save it as MP3:
  
  `ffmpeg -i video_filename -vn -ar 44100 -ac 2 -ab 192 -f mp3 sound.mp3`

- Convert frames from a video or GIF into individual numbered images:
  
  `ffmpeg -i video_or_gif_filename image%d.png`

- Combine numbered images (image1.jpg, image2.jpg, etc) into a video or GIF:
  
  `ffmpeg -f image2 -i image%d.jpg video.mpg_or_video.gif`

- Convert AVI video to MP4. AAC Audio @ 128kbit, Video @ 1250Kbit:
  
  `ffmpeg -i in.avi -acodec libfaac -ab 128k -vcodec mpeg4 -b 1250K out.mp4`

**fg**

Run jobs in foreground.

- Bring most recently suspended background job to foreground:
  
  `fg`

- Bring a specific job to foreground:
  
  `fg job_id`
**file**

Determine file type.

- Give a description of the type of the specified file. Works fine for files with no file extension:
  
  ```
  file filename
  ```

- Look inside a zipped file and determine the file type(s) inside:
  
  ```
  file -z foo.zip
  ```

- Allow file to work with special or device files:
  
  ```
  file -s filename
  ```

- Don’t stop at first file type match; keep going until the end of the file:
  
  ```
  file -k filename
  ```

- Determine the mime encoding type of a file:
  
  ```
  file -i filename
  ```

**find**

Find files under the given directory tree, recursively.

- Find files by extension:
  
  ```
  find root_path -name '*.ext'
  ```

- Find files matching path pattern:
  
  ```
  find root_path -path '*/lib/**/*.ext'
  ```

- Run a command for each file, use {} within the command to access the filename:
  
  ```
  find root_path -name '*.ext' -exec wc -l {} ;
  ```

- Find files modified since a certain time:
  
  ```
  find root_path -name '' -mtime -1d
  ```

- Find files using case insensitive name matching, of a certain size:
  
  ```
  find root_path -size +500k -size -10MB -iname '*.TaR.gZ'
  ```

- Delete files by name, older than a certain number of days:
  
  ```
  find root_path -name '*.ext' -mtime -180d -delete
  ```

- Find empty files or directories:
  
  ```
  find root_path -empty
  ```

- Find files matching more than one search criteria:
  
  ```
  find root_path -name '*.py' -or -name '*.r'
  ```
**fold**

Wraps each line in an input file to fit a specified width and prints it to the standard output.

- Wrap each line to default width (80 characters):
  
  ```bash
  fold file
  ```

- Wrap each line to width “30”:
  
  ```bash
  fold -w30 file
  ```

- Wrap each line to width “5” and break the line at spaces (puts each space separated word in a new line, words with length > 5 are wrapped):
  
  ```bash
  fold -w5 -s file
  ```

**for**

Shell loop over parameters.

- Perform a command with different arguments:
  
  ```bash
  for argument in 1 2 3; do command $argument; done
  ```

- Perform a command in every directory:
  
  ```bash
  for d in *; do (cd $d; command); done
  ```

**fortune**

Print a random quotation (fortune-cookie style).

- Print a quotation:
  
  ```bash
  fortune
  ```

- Print an offensive quotation:
  
  ```bash
  fortune -o
  ```

- Print a long quotation:
  
  ```bash
  fortune -l
  ```
- Print a short quotation:
  `fortune -s`
- List the available quotation database files:
  `fortune -f`
- Print a quotation from one of the database files listed by `fortune -f`:
  `fortune filename`

**fsck**

Check the integrity of a filesystem or repair it. The filesystem should be unmounted at the time the command is run.

- Check filesystem `/dev/sda`, reporting any damaged blocks:
  `fsck /dev/sda`
- Check filesystem `/dev/sda`, reporting any damaged blocks and interactively letting the user choose to repair each one:
  `fsck -r /dev/sda`
- Check filesystem `/dev/sda`, reporting any damaged blocks and automatically repairing them:
  `fsck -a /dev/sda`

**fswatch**

A cross-platform file change monitor.

- Run a bash command on file creation, update or deletion:
  `fswatch -0 path/to/file | xargs -0 bash_command`
- Watch one or more files and/or directories:
  `fswatch -0 path/to/file path/to/directory path/to/another_directory/**/*.js | xargs -0 bash_command`
- Use `{}` in your bash command as a placeholder for the absolute path to the changed file:
  `fswatch -0 path/to/directory | xargs -0 -I {} bash_command`
- Filter by event type, eg. Updated, Deleted or Created:
  `fswatch -0 --event Updated path/to/directory | xargs -0 bash_command`
**fswebcam**

Small and simple webcam for *nix.

- Take a picture:
  
  `fswebcam filename`

- Take a picture with custom resolution:
  
  `fswebcam -r widthxheight filename`

- Take a picture from selected device (Default is /dev/vidoe0):
  
  `fswebcam -d device filename`

- Take a picture with timestamp (timestamp string is formatted by strftime):
  
  `fswebcam --timestamp timestamp filename`

**ftp**

Tools to interact with a server via File Transfer Protocol.

- Connect to an FTP server:
  
  `ftp ftp.example.com`

- Switch to binary transfer mode (graphics, compressed files, etc):
  
  `binary`

- Transfer multiple files without prompting for confirmation on every file:
  
  `prompt off`

- Download multiple files (glob expression):
  
  `mget *.png`

- Upload multiple files (glob expression):
  
  `mput *.zip`

- Delete multiple files on the remote server:
  
  `mdelete *.txt`

- Rename a file on the remote server:
  
  `rename original_filename new_filename`
**fuck**

Corrects your previous console command.

- Set the `fuck` alias to `thefuck` tool:
  ```sh
eval "$(thefuck --alias)"
  ```
- Try to match a rule for the previous command:
  ```sh
  fuck
  ```

**fzf**

Command line fuzzy finder.

- Start finder on all files from arbitrary locations:
  ```sh
  find path/to/search -type f | fzf
  ```
- Start finder on running processes:
  ```sh
  ps axu | fzf
  ```
- Select multiple files with Shift-TAB and write to a file:
  ```sh
  find path/to/search_files -type f | fzf -m > filename
  ```
- Start finder with a given query:
  ```sh
  fzf -q "query"
  ```
- Start finder on entries that start with core and end with either go, rb, or py:
  ```sh
  fzf -q "^core go$ | rb$ | py$"
  ```
- Start finder on entries that not match pyc and match exactly travis:
  ```sh
  fzf -q "!pyc 'travis"
  ```
**gcc**

Preprocesses and compiles C and C++ source files, then assembles and links them together.

- Compile multiple source files into executable:
  
  ```
  gcc source1.c source2.c -o executable
  ```

- Allow warnings, debug symbols in output:
  
  ```
  gcc source.c -Wall -Og -o executable
  ```

- Include libraries from a different path:
  
  ```
  gcc source.c -o executable -Iheader_path -Llibrary_path -llibrary_name
  ```

- Compile source code into Assembler instructions:
  
  ```
  gcc -S source.c
  ```

- Compile source code without linking:
  
  ```
  gcc -c source.c
  ```

**gdb**

The GNU Debugger.

- Debug an executable:
  
  ```
  gdb executable
  ```

- Attach a process to gdb:
  
  ```
  gdb -p procID
  ```

- Execute given GDB commands upon start:
  
  ```
  gdb -ex "commands" executable
  ```

- Start gdb and pass arguments:
  
  ```
  gdb --args executable argument1 argument2
  ```
**gem**

Interact with the package manager for the Ruby programming language.

- Install latest version of a gem:
  
  ```bash
  gem install gemname
  ```

- Install specific version of a gem:
  
  ```bash
  gem install gemname -v 1.0.0
  ```

- Update a gem:
  
  ```bash
  gem update gemname
  ```

- List all gems:
  
  ```bash
  gem list
  ```

- Uninstall a gem:
  
  ```bash
  gem uninstall gemname
  ```

**ghc**

The Glasgow Haskell Compiler. Compiles and links Haskell source files.

- Find and compile all modules in the current directory:
  
  ```bash
  ghc Main
  ```

- Compile a single file:
  
  ```bash
  ghc file.hs
  ```

- Compile using extra optimization:
  
  ```bash
  ghc -O file.hs
  ```

- Stop compilation after generating object files (.o):
  
  ```bash
  ghc -c file.hs
  ```

- Run Haskell interactive interpreter (REPL):
  
  ```bash
  ghci
  ```
gifsicle

Create gifs.

- Make a GIF animation with gifsicle:
  
gifsicle --delay=10 --loop *.gif > anim.gif

- Extract frames from an animation:
  
gifsicle anim.gif '#0' > firstframe.gif

- You can also edit animations by replacing, deleting, or inserting frames:
  
gifsicle -b anim.gif --replace '#0' new.gif

git add

Adds changed files to the index.

- Add a file to the index:
  
git add path/to/file

- Add all files (tracked and untracked):
  
git add -A

- Only add already tracked files:
  
git add -u

- Also add ignored files:
  
git add -f

- Add parts of a file interactively:
  
git add -p path/to/file
**git bisect**

Use binary search to find the commit that introduced a bug. Git automatically jumps back and forth in the commit graph to progressively narrow down the faulty commit.

- Start a bisect session on a commit range bounded by a known buggy commit, and a known clean (typically older) one:

  `git bisect start bad_commit good_commit`

- For each commit that `git bisect` selects, mark it as "bad" or "good" after testing it for the issue:

  `git bisect good|bad`

- After `git bisect` pinpoints the faulty commit, end the bisect session and return to the previous branch:

  `git bisect reset`

- Skip a commit during a bisect (e.g. one that fails the tests due to a different issue):

  `git bisect skip`

**git blame**

Show commit hash and last author on each line of a file.

- Print file with author name and commit hash on each line:

  `git blame file`

- Print file with author email and commit hash on each line:

  `git blame -e file`
**git branch**

Main git command for working with branches.

- List local branches. The current branch is highlighted by *:
  
  `git branch`

- List all branches (local and remote):
  
  `git branch -a`

- Create new branch based on the current commit:
  
  `git branch branch_name`

- Rename a branch (must not have it checked out to do this):
  
  `git branch -m old_branch_name new_branch_name`

- Delete a local branch:
  
  `git branch -d branch_name`

**git checkout**

Checkout a branch or paths to the working tree.

- Create and switch to a new branch:
  
  `git checkout -b branch_name`

- Switch to an existing local or remote branch:
  
  `git checkout branch_name`

- Undo unstaged local modification:
  
  `git checkout .`

- Replace a file in the current working directory with the version of it committed in a given branch:
  
  `git checkout branch_name -- file_name`
**git cherry-pick**

Apply the changes introduced by existing commits to the current branch. To apply changes to another branch, first use `git-checkout` to switch to the desired branch.

- Apply a commit to the current branch:
  
  ```
git cherry-pick commit
  ```

- Apply a range of commits to the current branch (see also `git rebase --onto`):
  
  ```
git cherry-pick start_commit~..end_commit
  ```

- Apply multiple (non-sequential) commits to the current branch:
  
  ```
git cherry-pick commit_1 commit_2
  ```

**git clone**

Clone an existing repository.

- Clone an existing repository:
  
  ```
git clone remote_repository_location
  ```

- For cloning from the local machine:
  
  ```
git clone -l
  ```

- Do it quietly:
  
  ```
git clone -q
  ```

- Clone an existing repository, and truncate to the specified number of revisions, save your time mostly:
  
  ```
git clone --depth 10 remote_repository_location
  ```

**git commit**

Commit staged files to the repository.

- Commit staged files to the repository with comment:
  
  ```
git commit -m message
  ```

- Replace the last commit with currently staged changes:
  
  ```
git commit --amend
  ```
### git config

Get and set repository or global options.

- Print list of options for current repository:
  ```
git config --list --local
  ```

- Print global list of options, set in `~/.gitconfig`:
  ```
git config --list --global
  ```

- Get full list of options:
  ```
git config --list
  ```

- Get value of alias.ls option:
  ```
git config alias.st
  ```

- Set option alias.ls=status in file `~/.gitconfig`:
  ```
git config --global alias.ls "status"
  ```

- Remove option alias.st from `~/.gitconfig`:
  ```
git config --global --unset alias.st
  ```

### git diff

Show changes to tracked files.

- Show unstaged, uncommitted changes:
  ```
git diff
  ```

- Show all uncommitted changes (including staged ones):
  ```
git diff HEAD
  ```

- Show only staged (added, but not yet committed) changes:
  ```
git diff --staged
  ```

- Show only names of changed files since a given commit:
  ```
git diff --name-only commit
  ```

- Output a summary of file creations, renames and mode changes since a given commit:
  ```
git diff --summary commit
  ```

- Create a patch file:
  ```
git diff > target_file.patch
  ```
**git fetch**

Download objects and refs from a remote repository.

- Fetch the latest changes from the default remote upstream repository (if set):
  
  ```
  git fetch
  ```

- Fetch new branches from a specific remote upstream repository:
  
  ```
  git fetch remote_name
  ```

- Fetch the latest changes from all remote upstream repositories:
  
  ```
  git fetch --all
  ```

- Also fetch tags from the remote upstream repository:
  
  ```
  git fetch --tags
  ```

- Delete local references to remote branches that have been deleted upstream:
  
  ```
  git fetch --prune
  ```

**git-merge**

Perform a merge or rebase between two git branches incrementally. Conflicts between branches are tracked down to pairs of individual commits, to simplify conflict resolution.

- Start imerge-based rebase (checkout the branch to be rebased, first):
  
  ```
  git imerge rebase branch_to_rebase_on
  ```

- Start imerge-based merge (checkout the branch to merge into, first):
  
  ```
  git imerge merge branch_to_be_merged
  ```

- Show ASCII diagram of in-progress merge or rebase:
  
  ```
  git imerge diagram
  ```

- Continue imerge operation after resolving conflicts (git add the conflicted files, first):
  
  ```
  git imerge continue --no-edit
  ```

- Wrap up the imerge operation, after all conflicts are resolved:
  
  ```
  git imerge finish
  ```
**git init**

Initializes a new local Git repository.

- Initialize a new local repository:
  
  ```
git init
  ```

- Initialize a barebones repository, suitable for use as a remote over ssh:
  
  ```
git init --bare
  ```

**git log**

Show a history of commits.

- Show the sequence of commits starting from the current one, in reverse chronological order:
  
  ```
git log
  ```

- Show the history of a particular file or directory, including differences:
  
  ```
git log -p path
  ```

- Show only the first line of each commit message:
  
  ```
git log --oneline
  ```

- Show all commits, tags and branches for the entire repo in a graph format:
  
  ```
git log --oneline --decorate --all --graph
  ```

- Show only commits whose messages include a given string (case-insensitively):
  
  ```
git log -i --grep search_string
  ```

**git merge**

Merge branches.

- Merge a branch with your current branch:
  
  ```
git merge branch_name
  ```

- Edit the merge message:
  
  ```
git merge -e branch_name
  ```
**git mv**

Move or rename files and update the git index.

- Move file inside the repo and add the movement to the next commit:
  
  `git mv path/to/file new/path/to/file`

- Rename file and add renaming to the next commit:
  
  `git mv filename new_filename`

- Overwrite the file in the target path if it exists:
  
  `git mv --force file target`

**git pull**

Fetch branch from a remote repository and merge it to local repository.

- Download changes from default remote repository and merge it:
  
  `git pull`

- Download changes from default remote repository and use fast forward:
  
  `git pull --rebase`

- Download changes from given remote repository and branch, then merge them into HEAD:
  
  `git pull remote_name branch`

**git push**

Push commits to a remote repository.

- Send local changes in the current branch to its remote counterpart:
  
  `git push`

- Send local changes in a given branch to its remote counterpart:
  
  `git push remote_name local_branch`

- Publish the current branch to a remote repository, setting the remote branch name:

git push remote_name -u remote_branch

- Send changes on all local branches to their counterparts in a given remote repository:
  
git push --all remote_name

- Delete a branch in a remote repository:
  
git push remote_name --delete remote_branch

- Remove remote branches that don’t have a local counterpart:
  
git push --prune remote_name

- Publish tags that aren’t yet in the remote repository:
  
git push --tags

**git rebase**

Apply local commits on top of another branch’s history.

- Rebase your local branch interactively with the latest changes in local master:
  
git rebase -i master

- Rebase your local branch interactively with the latest changes from upstream:
  
git fetch origin; git rebase -i origin/master

- Handle an active rebase merge failure, after editing conflicting file(s):
  
git rebase --continue

- Abort a rebase in-progress:
  
git rebase --abort

- Rebase your local branch by specifying new base commit and old base commit:
  
git rebase --onto new_base_commit old_base_commit
**git remote**

Manage set of tracked repositories (“remotes”).
- Show a list of existing remotes, their names and URL:
  
git remote -v
- Add a remote:
  
git remote add remote_name remote_url
- Change the URL of a remote:
  
git remote set-url remote_name new_url
- Remove a remote:
  
git remote remove remote_name
- Rename a remote:
  
git remote rename old_name new_name

**git reset**

Undo commits or unstage changes, by resetting the current git HEAD to the specified state. If a path is passed, it works as “unstage”; if a commit hash or branch is passed, it works as “uncommit”.
- Unstage everything:
  
git reset
- Unstage specific file(s):
  
git reset path/to/file(s)
- Unstage portions of a file:
  
git reset -p path/to/file
- Undo the last commit, keeping its changes (and any further uncommitted changes) in the filesystem:
  
git reset HEAD~
- Undo the last two commits, adding their changes to the index, i.e. staged for commit:
  
git reset --soft HEAD~2
- Reset the repository to a given commit, discarding committed, staged and uncommitted changes since then:
  
git reset --hard commit
**git rm**

Remove files from repository index and local filesystem.

- Remove file from repository index and filesystem:
  
  `git rm file`

- Remove directory:
  
  `git rm -r directory`

- Remove file from repository index but keep it untouched locally:
  
  `git rm --cached file`

**git stash**

Stash local Git changes in a temporary area.

- Stash current changes, except new (untracked) files:
  
  `git stash [save optional_stash_message]`

- Stash current changes, including new (untracked) files:
  
  `git stash -u`

- Interactively select parts of changed files for stashing:
  
  `git stash -p`

- List all stashes (shows stash name, related branch and message):
  
  `git stash list`

- Apply a stash (default is the latest, named stash@{0}):  
  
  `git stash apply optional_stash_name_or_commit`

- Apply a stash (default is stash@{0}), and remove it from the stash list if applying doesn’t cause conflicts:
  
  `git stash pop optional_stash_name`

- Drop a stash (default is stash@{0}):
  
  `git stash drop optional_stash_name`

- Drop all stashes:
  
  `git stash clear`
**git status**

Show the index (changed files).
- Show changed files which are not yet added for commit:
  
  ```
  git status
  ```
- Give output in short format:
  
  ```
  git status -s
  ```

**git submodule**

Inspects, updates and manages submodules.
- Install a repository's specified submodules:
  
  ```
  git submodule update --init --recursive
  ```
- Add a git repository as a submodule:
  
  ```
  git submodule add repository_url
  ```
- Update every submodule to its latest commit:
  
  ```
  git submodule foreach git pull
  ```

**git svn**

Bidirectional operation between a Subversion repository and Git.
- Clone an SVN repository:
  
  ```
  git svn clone https://example.com/subversion_repo local_dir
  ```
- Clone a SVN repository starting at a given revision number:
  
  ```
  git svn clone -r 1234:HEAD https://svn.example.net/subversion/repo local_dir
  ```
- Update local clone from the remote SVN repository:
  
  ```
  git svn rebase
  ```
- Fetch updates from the remote SVN repository without changing the git HEAD:
  
  ```
  git svn fetch
  ```
- Commit back to the SVN repository:
  
  ```
  git svn dcommit
  ```
**git tag**

Create, list, delete or verify tags. A tag is a static reference to a specific commit.

- List all tags:
  
  `git tag`

- Create a tag with the given name pointing to the current commit:
  
  `git tag tag_name`

- Create an annotated tag with the given message:
  
  `git tag tag_name -m tag_message`

- Delete the tag with the given name:
  
  `git tag -d tag_name`

- Get updated tags from upstream:
  
  `git fetch --tags`

- List all tags whose ancestors include a given commit:
  
  `git tag --contains commit`

**git worktree**

Manage multiple working trees attached to the same repository.

- Create a new folder with the specified branch checked out into it:
  
  `git worktree add path/to/folder branch`

- Create a new folder with a new branch checked out into it:
  
  `git worktree add path/to/folder -b new_branch`

- List all the working directories attached to this repository:
  
  `git worktree list`

- Remove a worktree (after deleting worktree folder):
  
  `git worktree prune`
git

Distributed version control system.

- Check the Git version:
  
  `git --version`

- Call general help:
  
  `git --help`

- Call help on a command:
  
  `git help command`

- Execute Git command:
  
  `git command`


gitsome

A terminal-based interface for GitHub, accessed via the `gh` command. It also provides menu-style autocomplete suggestions for git commands.

- Enter the gitsome shell (optional), to enable autocompletion and interactive help for git (and gh) commands:
  
  `gitsome`

- Setup GitHub integration with the current account:
  
  `gh configure`

- List notifications for the current account (as would be seen in https://github.com/notifications):
  
  `gh notifications`

- List the current account's starred repos, filtered by a given search string:
  
  `gh starred "python 3"`

- View the recent activity feed of a given GitHub repository:
  
  `gh feed tldr-pages/tldr`

- View the recent activity feed for a given GitHub user, using the default pager (e.g. less):
  
  `gh feed torvalds -p`
**glances**

A cross-platform system monitoring tool.

- Run in terminal:
  ```
  glances
  ```
- Run in web server mode to show results in browser:
  ```
  glances -w
  ```
- Run in server mode to allow connections from other Glances clients:
  ```
  glances -s
  ```
- Connect to a Glances server:
  ```
  glances -c hostname
  ```
- Require a password in (web) server mode:
  ```
  glances -s --password
  ```

**go**

Tool for managing go source code.

- Download and install a package, specified by its import path:
  ```
  go get package_path
  ```
- Compile and run a source file (it has to contain a main package):
  ```
  go run file.go
  ```
- Compile the package present in the current directory:
  ```
  go build
  ```
- Execute all test cases of the current package (files have to end with _test.go):
  ```
  go test
  ```
- Compile and install the current package:
  ```
  go install
  ```
**gource**

Renders an animated tree diagram of Git, SVN, Mercurial and Bazaar repositories. It shows files and folders being created, modified or removed over time.

- Run gource in a directory (if it isn’t the repository’s root directory, the root is sought up from there):
  
gource path/to/repository

- Run gource in the current directory, with a custom output resolution:
  
gource -widthxheight

- Set a custom time scale for the animation:
  
gource -c time_scale_multiplier

- Set how long each day should be in the animation (this combines with -c, if provided):
  
gource -s seconds

- Set fullscreen mode and a custom background color:
  
gource -f -b hex_color_code

- Set a title for the animation:
  
gource --title title

**gpg**

Gnu Privacy Guard.

- Sign doc.txt without encryption (writes output to doc.txt.asc):
  
gpg --clearsign doc.txt

- Encrypt doc.txt for alice@example.com (output to doc.txt.gpg):
  
gpg --encrypt --recipient alice@example.com doc.txt

- Encrypt doc.txt with only a passphrase (output to doc.txt.gpg):
  
gpg --symmetric doc.txt
- Decrypt doc.txt.gpg (output to STDOUT):
  
gpg --decrypt doc.txt.gpg

- Import a public key:
  
gpg --import public.gpg

- Export public key for alice@example.com (output to STDOUT):
  
gpg --export --armor alice@example.com

- Export private key for alice@example.com (output to STDOUT):
  
gpg --export-secret-keys --armor alice@example.com

**gradle**

Gradle is the official build system for Android Studio.

- Compile a package:
  
grade build

- Clear the build folder:
  
grade clean

- Compile and Release package:
  
grade assembleRelease

**grep**

Matches patterns in input text. Supports simple patterns and regular expressions.

- Search for an exact string:
  
grep search_string path/to/file

- Search in case-insensitive mode:
  
grep -i search_string path/to/file
- Search recursively (ignoring non-text files) in current directory for an exact string:
  ```
grep -rI search_string .
  ```
- Use extended regular expressions (supporting ?, +, {}, ( ) and |):
  ```
grep -E ^regex$ path/to/file
  ```
- Print 3 lines of [C]ontext around, [B]efore, or [A]fter each match:
  ```
grep -C B A 3 search_string path/to/file
  ```
- Print the count of matches instead of the matching text:
  ```
grep -c search_string path/to/file
  ```
- Print line number for each match:
  ```
grep -n search_string path/to/file
  ```
- Print file names with matches:
  ```
grep -l search_string path/to/file
  ```
- Use the standard input instead of a file:
  ```
cat path/to/file | grep search_string
  ```
- Invert match for excluding specific strings:
  ```
grep -v search_string
  ```

**gulp**

JavaScript task runner and streaming build system. Tasks are defined within gulpfile.js at the project root.
- Run the default task:
  ```
gulp
  ```
- Run individual tasks:
  ```
gulp task othertask
  ```
**gunzip**

Extract file(s) from a gzip (.gz) archive.

- Extract a file from an archive, replacing the original file if it exists:
  
gunzip archive.tar.gz

- Extract a file to a target destination:
  
gunzip -c archive.tar.gz > archive.tar

- List the contents of a compressed file:
  
gunzip -l file.txt.gz

**gzip**

Compress/uncompress files with gzip compression (LZ77).

- Compress a file, replacing it with a gzipped compressed version:
  
gzip file.ext

- Decompress a file, replacing it with the original uncompressed version:
  
gzip -d file.ext.gz

- Compress a file specifying the output filename:
  
gzip -c file.ext > compressed_file.ext.gz

- Uncompress a gzipped file specifying the output filename:
  
gzip -c -d file.ext.gz > uncompressed_file.ext

- Specify the compression level. 1=Fastest (Worst), 9=Slowest (Best), Default level is 6:
  
gzip -9 -c file.ext > compressed_file.ext.gz
**handbrakecli**

Command-line interface to the HandBrake video conversion tool.

- Convert a video file to MKV (AAC 160kbit audio and x264 CRF20 video):
  
  ```
  handbrakecli -i input.avi -o output.mkv -e x264 -q 20 -B 160
  ```

- Resize a video file to 320x240:
  
  ```
  handbrakecli -i input.mp4 -o output.mp4 -w 320 -l 240
  ```

- List available presets:
  
  ```
  handbrakecli --preset-list
  ```

- Convert an AVI video to MP4 using the Android preset:
  
  ```
  handbrakecli --preset="Android" -i input.ext -o output.mp4
  ```

**haxelib**

Haxe Library Manager.

- Search for a Haxe library:
  
  ```
  haxelib search keyword
  ```

- Install a Haxe library:
  
  ```
  haxelib install libname
  ```

- Upgrade all installed Haxe libraries:
  
  ```
  haxelib upgrade
  ```

- Install the development version of a library from a Git repository:
  
  ```
  haxelib git libname git_url
  ```
**heroku**

Create and manage Heroku apps from the command line.

- Login to your heroku account:
  
  `heroku login`

- Create a heroku app:
  
  `heroku create`

- Show logs for an app:
  
  `heroku logs --app app_name`

- Run a one-off process inside a dyno (Heroku virtual machine):
  
  `heroku run process_name --app app_name`

- List dynos (Heroku virtual machines) for an app:
  
  `heroku ps --app app_name`

- Permanently destroy an app:
  
  `heroku destroy --app app_name`

**history**

Command Line history.

- Display the commands history list with line numbers:
  
  `history`

- Clear the commands history list (only for current bash shell):
  
  `history -c`

- Overwrite history file with history of current bash shell (often combined with `history -c` to purge history):
  
  `history -w`
hn

Command-line interface for Hacker News.

- View stories on Hacker News:
  
  hn

- View number of stories on Hacker News:
  
  hn --limit number

- View stories on Hacker News, and keep the list open after selecting a link:
  
  hn --keep-open

- View stories on Hacker News sorted by submission date:
  
  hn --latest

host

Lookup Domain Name Server.

- Lookup A, AAAA, and MX records of a domain:
  
  host domain

- Lookup a field (CNAME, TXT,...) of a domain:
  
  host -t field domain

- Reverse lookup an IP:
  
  host ip_address
htpasswd

Create and manage htpasswd files to protect web server directories using basic authentication.

- Create/overwrite htpasswd file:
  ```
  htpasswd -c path/to/file user_name
  ```
- Add user to htpasswd file or update existing user:
  ```
  htpasswd path/to/file user_name
  ```
- Add user to htpasswd file in batch mode without an interactive password prompt (for script usage):
  ```
  htpasswd -b path/to/file user_name password
  ```
- Delete user from htpasswd file:
  ```
  htpasswd -D path/to/file user_name
  ```
- Verify user password:
  ```
  htpasswd -v path/to/file user_name
  ```

http

HTTPie: HTTP client, a user-friendly cURL replacement.

- Download a URL to a file:
  ```
  http -d example.org
  ```
- Send form-encoded data:
  ```
  http -f example.org name='bob' profile_picture='bob.png'
  ```
- Send JSON object:
  ```
  http example.org name='bob'
  ```
- Specify an HTTP method:
  ```
  http HEAD example.org
  ```
- Include an extra header:
  ```
  http example.org X-MyHeader:123
  ```
- Pass a user name and password for server authentication:
  ```
  http -a username:password example.org
  ```
- Specify raw request body via stdin:
  ```
  cat data.txt | http PUT example.org
  ```
**hub**

A wrapper for git that adds commands for working with github-based projects. The commands can also be used using “git” instead of “hub”.

- Clone a repository you own, using just the repository name rather than the full URL:
  ```bash
  hub clone repo_name
  ```
- Clone another user’s repository, using their github username and the repository name:
  ```bash
  hub clone username/repo_name
  ```
- Create a fork of the current repository (cloned from another user) under your github profile:
  ```bash
  hub fork
  ```
- Create a PR of the current branch in the original repository (after pushing the branch to github):
  ```bash
  hub pull-request
  ```
- Upload the current (local-only) repository to your github account:
  ```bash
  hub create
  ```

**iconv**

Converts text from one encoding to another.

- Convert file to a specific encoding, and print to stdout:
  ```bash
  iconv -f from_encoding -t to_encoding input_file
  ```
- Convert file to the current locale’s encoding, and output to a file:
  ```bash
  iconv -f from_encoding input_file > output_file
  ```
- List supported encodings:
  ```bash
  iconv -l
  ```
id
Display current user and group identity.
- Display the current user identity as a number:
  \texttt{id -u}
- Display the current group identity as a number:
  \texttt{id -g}

if
Simple shell conditional.
- Echo a different thing depending on a command’s success:
  \texttt{command \&\& echo "success" || echo "failure"}
- Full if syntax:
  \texttt{if condition; then echo "true"; else echo "false"; fi}

ifconfig
Network Interface Configurator.
- View network settings of an ethernet adapter:
  \texttt{ifconfig eth0}
- Display details of all interfaces, including disabled interfaces:
  \texttt{ifconfig -a}
- Disable eth0 interface:
  \texttt{ifconfig eth0 down}
- Enable eth0 interface:
  \texttt{ifconfig eth0 up}
- Assign IP address to eth0 interface:
  \texttt{ifconfig eth0 \texttt{ip} \texttt{address}}
**import**

Capture some or all of an X server screen and save the image to a file. Part of ImageMagick library.

- Capture the entire X server screen in the Postscript image format:
  
  `import -window root output.postscript`

- Capture contents of remote x server screen in the png image format:
  
  `import -window root -display remote_host:{screen}.{display} output.png`

- Capture specific window with ID as displayed by xwininfo into jpg format:
  
  `import -window window_id output.jpg`

**in2csv**

Converts various tabular data formats into CSV. Included in csvkit.

- Convert an XLS file to CSV:
  
  `in2csv data.xls`

- Convert a DBF file to a CSV file:
  
  `in2csv data.dbf > data.csv`

- Convert a specific sheet from an XLSX file to CSV:
  
  `in2csv --sheet=sheet_name data.xlsx`

- Pipe a JSON file to in2csv:
  
  `cat data.json | in2csv -f json > data.csv`
**inkscape**

An SVG (Scalable Vector Graphics) editing program. Use -z to not open the GUI and only process files in the console.

- Open an SVG file in the Inkscape GUI:
  
  ```
  inkscape filename.svg
  ```

- Export an SVG file into a bitmap with the default format (PNG) and the default resolution (90 DPI):
  
  ```
  inkscape filename.svg -e filename.png
  ```

- Export an SVG file into a bitmap of 600x400 pixels (aspect ratio distortion may occur):
  
  ```
  inkscape filename.svg -e filename.png -w 600 -h 400
  ```

- Export a single object, given its ID, into a bitmap:
  
  ```
  inkscape filename.svg -i id -e object.png
  ```

- Export an SVG document to PDF, converting all texts to paths:
  
  ```
  inkscape filename.svg --export-pdf=filename.pdf --export-text-to-path
  ```

- Duplicate the object with id="path123", rotate the duplicate 90 degrees, save the file, and quit Inkscape:
  
  ```
  inkscape filename.svg --select=path123 --verb=EditDuplicate --verb=ObjectRotate90 --verb=FileSave --verb=FileQuit
  ```

**install**

Copy files and set attributes. Copy files (often executable) to a system location like `/usr/local/bin`, give them the appropriate permissions/ownership.

- Copy files to destination:
  
  ```
  install path/to/source path/to/destination
  ```

- Copy files to destination, setting their ownership:
  
  ```
  install -o user path/to/source path/to/destination
  ```

- Copy files to destination, setting their group ownership:
  
  ```
  install -g user path/to/source path/to/destination
  ```

- Copy files to destination, setting their mode:
  
  ```
  install -m +x path/to/source path/to/destination
  ```

- Copy files and apply access/modification times of source to destination:
  
  ```
  install -p path/to/source path/to/destination
  ```
ionice

Get or set program I/O scheduling class and priority. Scheduling classes: 1 (realtime), 2 (best-effort), 3 (idle). Priority levels: 0 (the highest) - 7 (the lowest).

- Set I/O scheduling class of a running process:
  ionice -c scheduling_class -p pid

- Run a command with custom I/O scheduling class and priority:
  ionice -c scheduling_class -n priority command

- Print the I/O scheduling class and priority of a running process:
  ionice -p pid

ioping

Monitor I/O latency in real time.

- Show disk I/O latency using the default values and the current directory:
  ioping .

- Measure latency on /tmp using 10 requests of 1 megabyte each:
  ioping -c 10 -s 1M /tmp

- Measure disk seek rate on /dev/sda:
  ioping -R /dev/sda

- Measure disk sequential speed on /dev/sda:
  ioping -RL /dev/sda

ipcs

Display information about resources used in IPC (Inter-process Communication).

- Specific information about the Message Queue which has the id 32768:
  ipcs -qi 32768

- General information about all the IPC:
  ipcs -a
**jar**

Java Applications/Libraries Packager.

- Unzip .jar/.war file to the current directory:
  
  ```
  jar -xvf *.jar
  ```

**java**

Java Application Launcher.

- Execute a java .class file that contains a main method by using just the class name:
  
  ```
  java filename
  ```

- Execute a .jar program:
  
  ```
  java -jar filename.jar
  ```

- Display JDK, JRE and HotSpot versions:
  
  ```
  java -version
  ```

**javac**

Java Application Compiler.

- Compile a .java file:
  
  ```
  javac filename.java
  ```

- Compile several .java files:
  
  ```
  javac filename1.java filename2.java filename3.java
  ```

- Compile all .java files in current directory:
  
  ```
  javac *.java
  ```

- Compile a .java file and place the resulting class file in a specific directory:
  
  ```
  javac -d path/to/some/directory filename.java
  ```
**jhat**

Java Heap Analysis Tool.
- Analyze a heap dump (from jmap), view via http on port 7000:
  ```
  jhat dump_file.bin
  ```
- Analyze a heap dump, specifying an alternate port for the http server:
  ```
  jhat -p port dump_file.bin
  ```
- Analyze a dump letting jhat use up to 8GB RAM (2-4x dump size recommended):
  ```
  jhat -J-mx8G dump_file.bin
  ```

**jmap**

Java Memory Map Tool.
- Print shared object mappings for a java process (output like pmap):
  ```
  jmap java_pid
  ```
- Print heap summary information:
  ```
  jmap -heap filename.jar java_pid
  ```
- Print histogram of heap usage by type:
  ```
  jmap -histo java_pid
  ```
- Dump contents of the heap into a binary file for analysis with jhat:
  ```
  jmap -dump:format=b,file=filename java_pid
  ```

**jobs**

Display status of jobs in the current session.
- Show status of all jobs:
  ```
  jobs
  ```
- Show status of a particular job:
  ```
  jobs job_id
  ```
- Show status and process IDs of all jobs:
  ```
  jobs -l
  ```
- Show process IDs of all jobs:
  ```
  jobs -p
  ```
**jq**

A lightweight and flexible command-line JSON processor.

- Output JSON file:
  
  ```bash
  cat file | jq
  ```

- Output all elements from JSON array in file, or all key-value pairs from JSON objects in file:
  
  ```bash
  cat file | jq []
  ```

- Read JSON objects from file, into array, and output (inverse of `jq []`):
  
  ```bash
  cat file | jq --slurp
  ```

- Output first element in JSON file:
  
  ```bash
  cat file | jq [0]
  ```

- Output "key" of first element in JSON file:
  
  ```bash
  cat file | jq [0].key
  ```

- Output "key" of each element in JSON file:
  
  ```bash
  cat file | jq 'map(.key)'
  ```

**jstack**

Java Stack Trace Tool.

- Print java stack traces for all threads in a java process:
  
  ```bash
  jstack java_pid
  ```

- Print mixed mode (java/c++) stack traces for all threads in a java process:
  
  ```bash
  jstack -m java_pid
  ```

- Print stack traces from java core dump:
  
  ```bash
  jstack /usr/bin/java file.core
  ```
**julia**

A high-level, high-performance dynamic programming language for technical computing.

- Start a Julia REPL session:
  
  julia

- Execute a Julia program and exit:
  
  julia program.jl

- Execute a Julia program that takes arguments:
  
  julia program.jl arguments

- Evaluate a string containing Julia code:
  
  julia -e 'julia_code'

- Evaluate a string of Julia code, passing arguments to it:
  
  julia -e 'for x in ARGS; println(x); end' arguments

- Start Julia in parallel mode, using N worker processes:
  
  julia -p N

**kill**

Sends a signal to a process, usually related to stopping the process. All signals except for SIGKILL and SIGSTOP can be intercepted by the process to perform a clean exit.

- Terminate a program using the default SIGTERM (terminate) signal:
  
  kill process_id

- List available signal names (to be used without the SIG prefix):
  
  kill -l

- Terminate a program using the SIGHUP (hang up) signal. Many daemons will reload instead of terminating:
kill -1|HUP process_id

- Terminate a program using the SIGINT (interrupt) signal. This is typically initiated by the user pressing Ctrl+C:

kill -2|INT process_id

- Signal the operating system to immediately terminate a program (which gets no chance to capture the signal):

kill -9|KILL process_id

- Signal the operating system to pause a program, it until a SIGCONT ("continue") signal is received:

kill -17|STOP process_id

last

View the last logged in users.

- View last logins, their duration and other information as read from /var/log/wtmp:

  last

- Specify how many of the last logins to show:

  last -n login_count

- View full login times and dates:

  last -F

- View the last login by a specific user:

  last user_name

- View the last reboot (last login of the pseudo user reboot):

  last reboot

- View the last shutdown (last login of the pseudo user shutdown):

  last shutdown
latexmk

Compile LaTeX source files into finished documents. Automatically does multiple runs when needed.

- Compile a dvi (DeVice Independent file) document from every source:
  latexmk

- Compile a dvi document from a specific source file:
  latexmk source.tex

- Compile a pdf document:
  latexmk -pdf source.tex

- Clean up all temporary tex files in the folder:
  latexmk -c

- Clean up temporary tex files created for a specific tex file:
  latexmk -c source.tex

- Clean up temporary tex and output files:
  latexmk -C

less

Open a file for interactive reading, allowing scrolling and search.

- Open a file:
  less source_file

- Page down / up:
  <Space> (down), b (up)

- Go to end / start of file:
  G (end), g (start)

- Forward search for a string (press n/N to go to next/previous match):
/something

- Backward search for a string (press n/N to go to next/previous match):
  
  ?something

- Open the current file in an editor:
  
  v

- Exit:
  
  q

**license**

Create license files for open-source projects.

- Print a license to stdout, using the defaults (auto-detected author name, and current year):
  
  license license_name

- Generate a license and save it to a file:
  
  license -o filename license_name

- List all available licenses:
  
  license ls

- Generate a license with custom author name and year:
  
  license --name author --year release_year license_name

**ln**

Creates links to files and folders.

- Create a symbolic link to a file (or folder):
  
  ln -s path/to/file path/to/symlink

- Overwrite an existing symbolic to point to a different file:
  
  ln -sf path/to/new_file path/to/symlink

- Create a hard link to a file:
  
  ln path/to/file path/to/hardlink
**logstash**

An ETL (extract, transform and load) tool. Commonly used to load data from various sources, like databases and log files, into elasticsearch.

- Check validity of a logstash configuration:
  ```
  logstash --configtest --config logstash_config.conf
  ```

- Run logstash using configuration:
  ```
  sudo logstash --config logstash_config.conf
  ```

- Run logstash with the most basic inline configuration string:
  ```
  sudo logstash -e 'input {} filter {} output {}'
  ```

**lp**

Print files.

- Print the output of a command to the default printer (see lpstat command):
  ```
  echo "test" | lp
  ```

- Print a file to the default printer:
  ```
  lp path/to/filename
  ```

- Print a file to a named printer (see lpstat command):
  ```
  lp -d printer_name path/to/filename
  ```

- Print N copies of file to default printer (replace N with desired number of copies):
  ```
  lp -n N path/to/filename
  ```

- Print only certain pages to the default printer (print pages 1, 3-5, and 16):
  ```
  lp -P 1,3-5,16 path/to/filename
  ```
**lpstat**

Show status information about printers.

- List printers present on the machine and whether they are enabled for printing:
  
  `lpstat -p`

- Show the default printer:
  
  `lpstat -d`

- Display all available status information:
  
  `lpstat -t`

- Show a list of print jobs queued by the specified user:
  
  `lpstat -u user`

**ls**

List directory contents.

- List files one per line:
  
  `ls -1`

- List all files, including hidden files:
  
  `ls -a`

- Long format list (permissions, ownership, size and modification date) of all files:
  
  `ls -la`

- Long format list with size displayed using human readable units (KB, MB, GB):
  
  `ls -lh`

- Long format list sorted by size (descending):
  
  `ls -lS`

- Long format list of all files, sorted by modification date (oldest first):
  
  `ls -ltr`
**ls/of**

Lists open files and the corresponding processes. Note: In most cases, you need root privilege (or use sudo) because you want to list files opened by others.

- Find the processes that have a given file open:
  ```
  lsof /path/to/file
  ```
- Find the process that opened a local internet port:
  ```
  lsof -i :port
  ```
- Only output the process PID:
  ```
  lsof -t /path/to/file
  ```
- List files opened by the given user:
  ```
  lsof -u username
  ```
- List files opened by the given command or process:
  ```
  lsof -c process_or_command_name
  ```
- List files opened by the given PID:
  ```
  lsof -p PID
  ```

**lwp-request**

Simple command-line HTTP client. Built with libwww-perl.

- Make a simple GET request:
  ```
  lwp-request -m GET http://example.com/some/path
  ```
- Upload a file with a POST request:
  ```
  cat /path/to/file | lwp-request -m POST http://example.com/some/path
  ```
- Make a request with a custom user agent:
  ```
  lwp-request -H 'User-Agent: user_agent' -m METHOD http://example.com/some/path
  ```
- Make a request with HTTP authentication:
  ```
  lwp-request -C username:password -m METHOD http://example.com/some/path
  ```
- Make a request and print request headers:
  ```
  lwp-request -U -m METHOD http://example.com/some/path
  ```
- Make a request and print response headers and status chain:
  ```
  lwp-request -E -m METHOD http://example.com/some/path
  ```
mailx

Send and receive mail.
- To send mail, the content is typed after the command and ended with Control-D:
  `mailx -s "subject" to_addr`
- Send mail with short content:
  `echo "content" | mailx -s "subject" to_addr`
- Send mail with content which written in a file:
  `mailx -s "subject" to_addr < content.txt`
- Send mail to a recipient and CC to another address:
  `mailx -s "subject" -c cc_addr to_addr`
- Send mail and set sender address:
  `mailx -s "subject" -r from_addr to_addr`
- Send mail with an attachment:
  `mailx -a file -s "subject" to_addr`

make

Task runner for targets described in Makefile. Mostly used to control the compilation of an executable from source code.
- Call the first target specified in the Makefile (usually named “all”):
  `make`
- Call a specific target:
  `make target`
- Use a specific Makefile:
  `make --file file`
- Execute make from another directory:
  `make --directory directory`
- Force making of a target, even if source files are unchanged:
  `make --always-make target`
**man**

Format and display manual pages.

- Display man page for a command:
  
  `man command`

- Display path searched for manpages:
  
  `man --path`

- Display location of a manpage rather than the manpage itself:
  
  `man -w command`

- Do a keyword search for manpages containing a search string:
  
  `man -k keyword`

**mdp**

A command-line based tool to make presentations from markdown files.

- Launch a presentation in the terminal from a markdown file:
  
  `mdp presentation.md`

- Disable fading transitions:
  
  `mdp --nofade presentation.md`

- Invert font colors to use in terminals with light background:
  
  `mdp --invert presentation.md`

- Disable transparency in transparent terminals:
  
  `mdp --notrans presentation.md`
**meshlabserver**

Command line interface for the MeshLab 3D mesh processing software.

- Convert an STL file to an OBJ file:
  ```
  meshlabserver -i input.stl -o output.obj
  ```

- Convert a WRL file to a OFF file, including the vertex and face normals in the output mesh:
  ```
  meshlabserver -i input.wrl -o output.off -om vn fn
  ```

- Dump a list of all the available processing filters into a file:
  ```
  meshlabserver -d filename
  ```

- Process a 3D file using a filter script created in the MeshLab GUI (Filters > Show current filter script > Save Script):
  ```
  meshlabserver -i input.ply -o output.ply -s filter_script.mlx
  ```

- Process a 3D file using a filter script, writing the output of the filters into a log file:
  ```
  meshlabserver -i input.x3d -o output.x3d -s filter_script.mlx -l logfile
  ```

**meteor**

Full-stack javascript platform for building web applications.

- Run a meteor project from its root directory in development mode:
  ```
  meteor
  ```

- Create a project under the given directory:
  ```
  meteor create path/to/directory
  ```

- Display the list of packages the project is currently using:
  ```
  meteor list
  ```

- Add a package to the project:
  ```
  meteor add package_name
  ```

- Remove a package from the project:
  ```
  meteor remove package_name
  ```

- Create a production build of the project as a tarball under the given directory:
  ```
  meteor build path/to/directory
  ```
mitmdump

View, record, and programatically transform HTTP traffic. The command-line counterpart to mitmproxy.

- Start a proxy and save all output to a file:
  
  mitmdump -w filename

- Filter a saved traffic file to just POST requests:
  
  mitmdump -nr input_filename -w output_filename "~m post"

- Replay a saved traffic file:
  
  mitmdump -nc filename

mitmproxy

An interactive man-in-the-middle HTTP proxy.

- Start mitmproxy with default settings:
  
  mitmproxy

- Start mitmproxy bound to custom address and port:
  
  mitmproxy -b ip_address -p port

mkdir

Creates a directory.

- Create a directory in current folder or given path:
  
  mkdir directory

- Create directories recursively (useful for creating nested dirs):
  
  mkdir -p path/to/directory
mkfifo

Makes FIFOs (named pipes).
- Create a named pipe at a given path:
  mkfifo path/to/pipe

mmv

Move and rename files in bulk.
- Rename all files with a certain extension to a different extension:
  mmv "*.old_extension" "#1.new_extension"
- Copy report6part4.txt to ./french/rapport6partie4.txt along with all similarly named files:
  mmv -c "report*part*.txt" "./french/rapport#1partie#2.txt"
- Append all .txt files into one file:
  mmv -a "*.txt" "all.txt"
- Convert dates in filenames from "M-D-Y" format to "D-M-Y" format:
  mmv "[0-1][0-9]-[0-3][0-9]-[0-9][0-9][0-9][0-9].txt" "#3#4-#1#2-#5#6#7#8.txt"

mocha

Execute Mocha JavaScript test runner.
- Run tests with default configuration or as configured in mocha.opts:
  mocha
- Run tests contained at a specific location:
  mocha folder/with/tests
- Run tests that match a specific grep pattern:
  mocha --grep ^regex$
- Run tests on changes to JavaScript files in the current directory and once initially:
  mocha --watch
- Run tests with a specific reporter:
  mocha --reporter reporter
**mongo**

MongoDB interactive shell client.

- Connect to a database:
  
  `mongo database`

- Connect to a database running on a given host on a given port:
  
  `mongo --host host --port port database`

- Connect to a database with a given username; user will be prompted for password:
  
  `mongo --username username database --password`

- Evaluate a javascript expression on the database:
  
  `mongo --eval 'JSON.stringify(db.foo.findOne())' database`

**mongodump**

Utility to export the contents of a MongoDB instance.

- Create a dump of all databases (this will place the files inside a folder called "dump"):
  
  `mongodump`

- Specify an output location for the dump:
  
  `mongodump --out path/to/folder`

- Create a dump of a given database:
  
  `mongodump --db database_name`

- Create a dump of a given collection within a given database:
  
  `mongodump --collection collection_name --db database_name`

- Connect to a given host running on a given port, and create a dump:
  
  `mongodump --host host --port port`

- Create a dump of a given database with a given username; user will be prompted for password:
  
  `mongodump --username username database --password`
**mongorestore**

Utility to import a collection or database from a binary dump into a MongoDB instance.

- Import a bson data dump from a folder to a MongoDB database:
  ```
  mongorestore --db database_name path/to/folder
  ```

- Import a bson data dump from a folder to a given database in a MongoDB server host, running at a given port, with user authentication (user will be prompted for password):
  ```
  mongorestore --host database_host:port --db database_name --username username path/to/folder --password
  ```

- Import a collection from a bson file to a MongoDB database:
  ```
  mongorestore --db database_name path/to/file
  ```

- Import a collection from a bson file to a given database in a MongoDB server host, running at a given port, with user authentication (user will be prompted for password):
  ```
  mongorestore --host database_host:port --db database_name --username username path/to/file --password
  ```

**montage**

Imagemagick imagemontage tool. Tiles images into a customisable grid.

- Tile images into a grid, automatically resizing images larger than the grid cell size:
  ```
  montage image1.png image2.jpg imageN.png montage.jpg
  ```

- Tile images into a grid, automatically calculating the grid cell size from the largest image:
  ```
  montage image1.png image2.jpg imageN.png -geometry +0+0 montage.jpg
  ```

- Set the grid cell size and resize images to fit it before tiling:
  ```
  montage image1.png image2.jpg imageN.png -geometry 640x480+0+0 montage.jpg
  ```

- Limit the number of rows and columns in the grid, causing input images to overflow into multiple output montages:
  ```
  montage image1.png image2.jpg imageN.png -geometry +0+0 -tile 2x3 montage_%d.jpg
  ```

- Resize and crop images to completely fill their grid cells before tiling:
  ```
  montage image1.png image2.jpg imageN.png -geometry +0+0 -resize 640x480^ -gravity center -crop 640x480+0+0 montage.jpg
  ```
**more**

Open a file for interactive reading, allowing scrolling and search (in forward direction only).

- Open a file:
  
  `more source_file`

- Page down:
  
  `<Space>`

- Search for a string (press n to go to the next match):
  
  `/something`

- Exit:
  
  `q`

**mount**

Provides access to an entire filesystem in one directory.

- Show all mounted filesystems:
  
  `mount`

- Mount a device to a directory:
  
  `mount -t filesystem_type path/to/device_file path/to/target_directory`

- Mount a CD-ROM device (with the filetype ISO9660) to /cdrom (readonly):
  
  `mount -t iso9660 -o ro /dev/cdrom /cdrom`

- Mount all the filesystem defined in /etc/fstab:
  
  `mount -a`

- Mount a specific filesystem described in /etc/fstab (e.g. "/dev/sda1 /my_drive ext2 defaults 0 2"):
  
  `mount /my_drive`
**mp4box**

MPEG-4 Systems Toolbox - Muxes streams into MP4 container.

- Display information about an existing MP4 file:
  
  ```
  mp4box -info filename
  ```

- Add an SRT subtitle file into an MP4 file:
  
  ```
  mp4box -add input_subs.srt:lang=eng -add input.mp4 output.mp4
  ```

- Combine audio from one file and video from another:
  
  ```
  mp4box -add input1.mp4#audio -add input2.mp4#video output.mp4
  ```

**mpc**

Music Player Client. Program for controlling the Music Player Daemon (MPD).

- Toggle play/pause:
  
  ```
  mpc toggle
  ```

- Stop playing:
  
  ```
  mpc stop
  ```

- Show information about the currently playing song:
  
  ```
  mpc status
  ```

- Play next song:
  
  ```
  mpc next
  ```

- Play previous song:
  
  ```
  mpc prev
  ```

- Forward or rewind the currently playing song:
  
  ```
  mpc [+-]seconds
  ```
**msmtp**

An SMTP client. It reads text from standard input and sends it to an SMTP server.

- Send an email using the default account configured in ~/.msmtprc:

  ```
  echo "Hello world" | msmtp to@example.org
  ```

- Send an email using a specific account configured in ~/.msmtprc:

  ```
  echo "Hello world" | msmtp --account=account_name to@example.org
  ```

- Send an email without a configured account. The password should be specified in the ~/.msmtprc file:

  ```
  echo "Hello world" | msmtp --host=localhost --port=999 --from=from@example.org to@example.org
  ```

**mtr**

Matt's Traceroute: combined traceroute and ping tool.

- Traceroute to a host and continuously ping all intermediary hops:

  ```
  mtr host
  ```

- Disable IP address and host name mapping:

  ```
  mtr -n host
  ```

- Generate output after pinging each hop 10 times:

  ```
  mtr -w host
  ```

- Force IP IPv4 or IPV6:

  ```
  mtr -4 host
  ```
mv

Move or rename files and directories.

- Move files in arbitrary locations:
  
  `mv source target`

- Do not prompt for confirmation before overwriting existing files:
  
  `mv -f source target`

- Do not prompt for confirmation before overwriting existing files but write to standard error before overriding:
  
  `mv -fi source target`

- Move files in verbose mode, showing files after they are moved:
  
  `mv -v source target`

mysql

The MySQL command-line tool.

- Connect to a database:
  
  `mysql database_name`

- Connect to a database, user will be prompted for a password:
  
  `mysql -u user --password database_name`

- Connect to a database on another host:
  
  `mysql -h database_host database_name`

- Execute SQL statements in a script file (batch file):
  
  `mysql database_name < script.sql`
**mysqldump**

Backups mysql databases.

- Create a backup, user will be prompted for a password:
  
  `mysqldump -u user --password database_name > filename.sql`

- Restore a backup, user will be prompted for a password:
  
  `mysql -u user --password database_name < filename.sql`

**nano**

Simple, easy to use editor. An enhanced, free Pico clone.

- Start nano in terminal with `{filename}`:
  
  `nano filename`

- Enable smooth scrolling:
  
  `nano -S filename`

- Indent new lines to the previous lines’ indentation:
  
  `nano -i filename`

**nc**

Reads and writes tcp or udp data.

- Listen on a specified port:
  
  `nc -l port`

- Connect to a certain port (you can then write to this port):
  
  `nc ip_address port`

- Set a timeout:
  
  `nc -w timeout_in_seconds ipaddress port`

- Serve a file:
nc -l port < file

- Receive a file:

  nc ip_address port > file

- Server stay up after client detach:

  nc -k -l port

- Client stay up after EOF:

  nc -q timeout ip_address

- Port scanning:

  nc -v -z ip_address port

- Proxy and port forwarding:

  nc -l port | nc hostname port

**nginx**

Nginx web server.

- Start server with default config:

  nginx

- Start server with custom config file:

  nginx -c config_file

- Start server with a prefix for all relative paths in config file:

  nginx -c config_file -p prefix/for/relative/paths

- Test configuration without affecting the running server:

  nginx -t

- Reload configuration by sending a signal with no downtime:

  nginx -s reload
**ngrep**

Filter network traffic packets using regular expressions.

- Capture traffic of all interfaces:
  
  ngrep -d any

- Capture traffic of a specific interface:
  
  ngrep -d eth0

- Capture traffic crossing port 22 of interface eth0:
  
  ngrep -d eth0 port 22

- Capture traffic from or to a host:
  
  ngrep host www.example.com

- Filter keyword 'User-Agent:' of interface eth0:
  
  ngrep -d eth0 'User-Agent:'

**nice**

Execute a program with a custom scheduling priority (niceness). Niceness values range from -20 (the highest priority) to 19 (the lowest).

- Launch a program with altered priority:
  
  nice -n niceness_value command

**nix-env**

Manipulate or query Nix user environments.

- Show available package with name or without name:
  
  nix-env -qa pkg_name

- Show the status of available packages:
  
  nix-env -qas
- Install package:
  ```
nix-env -i pkg_name
  ```
- Uninstall package:
  ```
nix-env -e pkg_name
  ```
- Upgrade one package:
  ```
nix-env -u pkg_name
  ```
- Upgrade all packages:
  ```
nix-env -u
  ```

**nmap**

Network exploration tool and security / port scanner. Some features only activate when Nmap is run with privileges.

- Try to determine whether the specified hosts are up and what are their names:
  ```
nmap -sn ip_or_hostname optional_another_address
  ```
- Like above, but also run a default 1000-port TCP scan if host seems up:
  ```
nmap ip_or_hostname optional_another_address
  ```
- Also enable scripts, service detection, OS fingerprinting and traceroute:
  ```
nmap -A address_or_addresses
  ```
- Assume good network connection and speed up execution:
  ```
nmap -T4 address_or_addresses
  ```
- Scan a specific list of ports (use -p- for all ports 1-65535):
  ```
nmap -p port1,port2,...,portN address_or_addresses
  ```
- Perform TCP and UDP scanning (use -SU for UDP only, -sZ for SCTP, -sO for IP):
  ```
nmap -SSU address_or_addresses
  ```
**node**

Server-side JavaScript platform (Node.js).

- Run a JavaScript file:
  ```
  node file.js
  ```

- Start a REPL (interactive shell):
  ```
  node
  ```

- Evaluate JavaScript by passing it in the command:
  ```
  node -e "code"
  ```

**nohup**

Allows for a process to live when the terminal gets killed.

- Run process that can live beyond the terminal:
  ```
  nohup command options
  ```

**npm**

JavaScript and Node.js package manager. Manage Node.js projects and their module dependencies.

- Download and install a module globally:
  ```
  npm install -g module_name
  ```

- Download all dependencies referenced in package.json:
  ```
  npm install
  ```

- Download a given dependency required for the application to run, and add it to the package.json:
  ```
  npm install module_name@version --save
  ```

- Download a given dependency for development purposes, and add it to the package.json:
  ```
  npm install module_name@version --save-dev
  ```
- Uninstall a module:
  
  `npm uninstall module_name`

- List a tree of installed modules:
  
  `npm list`

- Interactively create a package.json file:
  
  `npm init`

**nslookup**

Query name server(s) for various domain records.

- Query your system’s default name server for an IP address (A record) of the domain:
  
  `nslookup example.com`

- Query a given name server for a NS record of the domain:
  
  `nslookup -type=NS example.com 8.8.8.8`

- Query for a reverse lookup (PTR record) of an IP address:
  
  `nslookup -type=PTR 54.240.162.118`

- Query for ANY available records using TCP protocol:
  
  `nslookup -vc -type=ANY example.com`

- Query a given name server for the whole zone file (zone transfer) of the domain using TCP protocol:
  
  `nslookup -vc -type=AXFR example.com name_server`

- Query for a mail server (MX record) of the domain, showing details of the transaction:
  
  `nslookup -type=MX -debug example.com`

- Query a given name server on a specific port number for a TXT record of the domain:
  
  `nslookup -port=port_number -type=TXT example.com name_server`
nvm

Node.js version manager. Switch between NodeJS versions: system, node, 0.10, 0.12, 4.2 etc.

- Install a specific version of NodeJS:
  
  nvm install node_version

- Use a specific version NodeJS in the current shell:
  
  nvm use node_version

- Set the default NodeJS version:
  
  nvm alias default node_version

- List all available NodeJS versions and print the default one:
  
  nvm list

- Run a specific version NodeJS REPL:
  
  nvm run node_version --version

- Run app in a specific version of NodeJS:
  
  nvm exec node_version node app.js

odps auth

User authorities in ODPS (Open Data Processing Service).

- Add a user to the current project:
  
  add user user_name;

- Grant a set of authorities to a user:
  
  grant action_list on object_type object_name to user user_name;

- Show authorities of a user:
  
  show grants for user_name;

- Create a user role:
create role role_name;
- Grant a set of authorities to a role:
  grant action_list on object_type object_name to role role_name;
- Describe authorities of a role:
  desc role role_name;
- Grant a role to a user:
  grant role_name to user_name;

**odps func**

Manage functions in ODPS (Open Data Processing Service).
- Show functions in the current project:
  list functions;
- Create a Java function using a .jar resource:
  create function func_name as path.to.package.Func using 'package.jar';
- Create a Python function using a .py resource:
  create function func_name as script.Func using 'script.py';
- Delete a function:
  drop function func_name;

**odps inst**

Manage instances in ODPS (Open Data Processing Service).
- Show instances created by current user:
  show instances;
- Describe the details of an instance:
  desc instance instance_id;
- Check the status of an instance:
  status instance_id;
- Wait on the termination of an instance, printing log and progress information until then:
  wait instance_id;
- Kill an instance:
  kill instance_id;
**odps resource**

Manage resources in ODPS (Open Data Processing Service).

- Show resources in the current project:
  
  list resources;

- Add file resource:
  
  add file file_name as alias;

- Add archive resource:
  
  add archive archive.tar.gz as alias;

- Add .jar resource:
  
  add jar package.jar;

- Add .py resource:
  
  add py script.py;

- Delete resource:
  
  drop resource resource_name;

**odps table**

Create and modify tables in ODPS (Open Data Processing Service).

- Create a table with partition and lifecycle:
  
  create table table_name (col type) partitioned by (col type) lifecycle days;

- Create a table based on the definition of another table:
  
  create table table_name like another_table;

- Add partition to a table:
  
  alter table table_name add partition (partition_spec);

- Delete partition from a table:
  
  alter table table_name drop partition (partition_spec);

- Delete table:
  
  drop table table_name;
**odps tunnel**

Data tunnel in ODPS (Open Data Processing Service).

- Download table to local file:
  
  ```
  tunnel download table_name file;
  ```

- Upload local file to a table partition:
  
  ```
  tunnel upload file table_name/partition_spec;
  ```

- Upload table specifying field and record delimiters:
  
  ```
  tunnel upload file table_name -fd field_delim -rd record_delim;
  ```

- Upload table using multiple threads:
  
  ```
  tunnel upload file table_name -threads num;
  ```

**odps**

Aliyun ODPS (Open Data Processing Service) command line tool.

- Start the command line with a custom configuration file:
  
  ```
  odpscmd --config=odps_config.ini
  ```

- Switch current project:
  
  ```
  use project_name;
  ```

- Show tables in the current project:
  
  ```
  show tables;
  ```

- Describe a table:
  
  ```
  desc table_name;
  ```

- Show table partitions:
  
  ```
  show partitions table_name;
  ```

- Describe a partition:
  
  ```
  desc table_name partition (partition_spec);
  ```
**openssl**

OpenSSL cryptographic toolkit.

- Generate a 2048bit RSA private key and save it to a file:

  openssl genrsa -out filename.key 2048

- Generate a certificate signing request to be sent to a certificate authority:

  openssl req -new -sha256 -key filename.key -out filename.csr

- Generate a self-signed certificate from a certificate signing request valid for some number of days:

  openssl x509 -req -days days -in filename.csr -signkey filename.key -out filename.crt

- Display the certificate presented by an SSL/TLS server:

  openssl s_client -connect host:port </dev/null

- Display the complete certificate chain of an HTTPS server:

  openssl s_client -connect host:443 -showcerts </dev/null

**optipng**

PNG image file optimization utility.

- Compress a PNG with default settings:

  optipng file.png

- Compress a PNG with best compression:

  optipng -o7 file.png

- Compress a PNG with fastest compression:

  optipng -o8 file.png

- Compress a PNG and add interlacing:

  optipng -i 1 file.png

- Compress a PNG and remove all metadata:

  optipng -strip all file.png
**pandoc**

Convert documents between various formats.

- Convert file to pdf (the output format is automatically determined from the output file’s extension):
  
pandoc input.md -o output.pdf

- Convert a file to a specific output format (useful for when the extension alone is ambiguous):
  
pandoc input.docx --to markdown_github -o output.md

- List all supported input formats:
  
pandoc --list-input-formats

- List all supported output formats:
  
pandoc --list-output-formats

**parallel**

Run commands on multiple CPU cores.

- Gzip several files at once, using all cores:
  
parallel gzip ::: file1 file2 file3

- Read arguments from stdin, run 4 jobs at once:
  
ls *.txt | parallel -j4 gzip

- Convert JPG images to PNG using replacement strings:
  
parallel convert {} {}.png ::: *.jpg

- Parallel xargs, cram as many args as possible onto one command:
  
args | parallel -X command

- Break stdin into ~1M blocks, feed each block to stdin of new command:
  
cat big_file.txt | parallel --pipe --block 1M command

- Run on multiple machines via SSH:
  
parallel -S machine1,machine2 command ::: arg1 arg2
pass

Safely store and read passwords or other sensitive data easily. All data is GPG-encrypted, and managed with a git repository.

- Initialize the storage using a gpg-id for encryption:
  pass init gpg_id

- Save a new password (prompts you for the value without echoing it):
  pass insert path/to/data

- Copy a password (first line of the data file) to the clipboard:
  pass -c path/to/data

- List the whole store tree:
  pass

- Generate a new random password with a given length, and copy it to the clipboard:
  pass generate -c path/to/data num

- Run any git command against the underlying store repository:
  pass git git_arguments

passwd

Passwd is a tool used to change a user's password.

- Change the password of the current user:
  passwd new password

- Change the password of the specified user:
  passwd username new password

- Get the current status of the user:
  passwd -S

- Make the password of the account blank (it will set the named account passwordless):
  passwd -d
**paste**

Merge lines of files.

- Join all the lines into a single line, using TAB as delimiter:
  
  `paste -s file`

- Join all the lines into a single line, using the specified delimiter:
  
  `paste -s -d delimiter file`

- Merge two files side by side, each in its column, using TAB as delimiter:
  
  `paste file1 file2`

- Merge two files side by side, each in its column, using the specified delimiter:
  
  `paste -d delimiter file1 file2`

- Merge two files, with lines added alternatively:
  
  `paste -d '\n' file1 file2`

**patch**

Patch a file (or files) with a diff file. Note that diff files contain both the target filenames and list of changes.

- Apply a patch:
  
  `patch < patch_file.diff`

- Apply a patch to current directory:
  
  `patch -p1 < patch_file.diff`

- Apply the reverse of a patch:
  
  `patch -R < patch_file.diff`
**pdflatex**

Compile a pdf document from LaTeX source files.

- Compile a pdf document:
  
  \texttt{pdflatex source.tex}

- Compile a pdf document, halting on each error:
  
  \texttt{pdflatex -halt-on-error source.tex}

**pdftk**

PDF toolkit.

- Extract pages 1-3, 5 and 6-10 from a PDF file and save them as another one:
  
  \texttt{pdftk input.pdf cat 1-3 5 6-10 output output.pdf}

- Merge (concatenate) a list of PDF files and save the result as another one:
  
  \texttt{pdftk file1.pdf file2.pdf ... cat output output.pdf}

- Split each page of a PDF file into a separate file, with a given filename output pattern:
  
  \texttt{pdftk input.pdf burst output out\_%d.pdf}

- Rotate all pages by 180 degrees clockwise:
  
  \texttt{pdftk input.pdf cat 1-endsouth output output.pdf}

- Rotate third page by 90 degrees clockwise and leave others unchanged:
  
  \texttt{pdftk input.pdf cat 1-2 3east 4-end output output.pdf}
**perl**

The Perl 5 language interpreter.

- Parse and execute a Perl script:
  
  ```
  perl script.pl
  ```

- Check syntax errors on a Perl script:
  
  ```
  perl -c script.pl
  ```

- Parse and execute a perl statement:
  
  ```
  perl -e perl_statement
  ```

- Import module before execution of a perl statement:
  
  ```
  perl -Mmodule -e perl_statement
  ```

- Run a Perl script in debug mode, using perldebug:
  
  ```
  perl -d script.pl
  ```

**pg_dump**

Extract a PostgreSQL database into a script file or other archive file.

- Dump database into a SQL-script file:
  
  ```
  pg_dump db_name > output_file.sql
  ```

- Same as above, customize username:
  
  ```
  pg_dump -U username db_name > output_file.sql
  ```

- Same as above, customize host and port:
  
  ```
  pg_dump -h host -p port db_name > output_file.sql
  ```

- Dump a database into a custom-format archive file:
  
  ```
  pg_dump -Fc db_name > output_file.dump
  ```
pg_restore

Restore a PostgreSQL database from an archive file created by pg_dump.

- Restore an archive into an existing database:
  
  `pg_restore -d db_name archive_file.dump`

- Same as above, customize username:
  
  `pg_restore -U username -d db_name archive_file.dump`

- Same as above, customize host and port:
  
  `pg_restore -h host -p port -d db_name archive_file.dump`

- Clean database objects before creating them:
  
  `pg_restore --clean -d db_name archive_file.dump`

- Use multiple jobs to do the restoring:
  
  `pg_restore -j 2 -d db_name archive_file.dump`

pgrep

Find or signal process by name.

- Return PIDs of any running processes with a matching command string:
  
  `pgrep process_name`

- Search full command line with parameters instead of just the process name:
  
  `pgrep -f "process_name parameter"`

- Search for process run by a specific user:
  
  `pgrep -u root process_name`
php

PHP command line interface.

- Parse and execute a php script:
  `php file`

- Check syntax on (i.e. lint) a PHP script:
  `php -l file`

- Run PHP interactively:
  `php -a`

- Run PHP code (Notes: Don’t use <? ?> tags; escape double quotes with backslash):
  `php -r "code"

- Start a PHP built-in web server in the current directory:
  `php -S host:port`

phpize

Prepare a PHP extension for compiling.

- Prepare the PHP extension in the current directory for compiling:
  `phpize`

- Delete files previously created by phpize:
  `phpize --clean`

phpunit

PHPUnit command-line test runner.

- Run tests in the current directory. Note: Expects you to have a ‘phpunit.xml’:
  `phpunit`

- Run tests in a specific file:
  `phpunit path/to/TestFile.php`

- Run tests annotated with the given group:
  `phpunit --group name`

- Run tests and generate a coverage report in HTML:
  `phpunit --coverage-html directory`
**pigz**

Multithreaded zlib compression utility.

- Compress a file with default options:
  
  ```bash
  pigz filename
  ```

- Compress a file using the best compression method:
  
  ```bash
  pigz -9 filename
  ```

- Compress a file using no compression and 4 processors:
  
  ```bash
  pigz -0 -p4 filename
  ```

- Decompress a file:
  
  ```bash
  pigz -d archive.gz
  ```

- List the contents of an archive:
  
  ```bash
  pigz -l archive.tar.gz
  ```

**ping**

Send ICMP ECHO_REQUEST packets to network hosts.

- Ping host:
  
  ```bash
  ping host
  ```

- Ping a host only a specific number of times:
  
  ```bash
  ping -c count host
  ```

- Ping host, specifying the interval in seconds between requests (default is 1 second):
  
  ```bash
  ping -i seconds host
  ```

- Ping host without trying to lookup symbolic names for addresses:
  
  ```bash
  ping -n host
  ```

- Ping host and ring the bell when a packet is received (if your terminal supports it):
  
  ```bash
  ping -a host
  ```
**pip**

Python package manager.

- Install a package:
  
  ```
  pip install package_name
  ```

- Install a specific version of a package:
  
  ```
  pip install package_name==package_version
  ```

- Upgrade a package:
  
  ```
  pip install -U package_name
  ```

- Uninstall a package:
  
  ```
  pip uninstall package_name
  ```

- Save installed packages to file:
  
  ```
  pip freeze > requirements.txt
  ```

- Install packages from file:
  
  ```
  pip install -r requirements.txt
  ```

**pkill**

Signal process by name. Mostly used for stopping processes.

- Kill all processes which match:
  
  ```
  pkill -9 process_name
  ```

- Kill all processes which match their full command instead of just the process name:
  
  ```
  pkill -9 -f "command_name"
  ```

- Send SIGUSR1 signal to processes which match:
  
  ```
  pkill -USR1 process_name
  ```
**play**

Audio player of SoX - Sound eXchange. Plays any audio from the command line, with audio formats identified by the extension.

- Play the given audio file:
  
  `play audiofile`

- Play the given audio files:
  
  `play audiofile1 audiofile2`

- Play the given audio at twice the speed:
  
  `play audiofile speed 2.0`

- Play the given audio in reverse:
  
  `play audiofile reverse`

**pngcrush**

PNG image compression utility.

- Compress a PNG file:
  
  `pngcrush in.png out.png`

- Compress all PNGs and output to directory:
  
  `pngcrush -d path/to/output *.png`

- Compress PNG file with all 114 available algorithms and pick the best result:
  
  `pngcrush -rem allb -brute -reduce in.png out.png`
printf

Format and print text.

- Print a text message:
  
  printf \"%s\n\" "Hello world"

- Print an integer in bold blue:
  
  printf \"\e[1;34m%3d\e[0m\n\" 42

- Print a float number with the unicode Euro sign:
  
  printf \"\u20AC %.2f\n\" 123.4

- Print a text message composed with environment variables:
  
  printf \"var1: %s\ntvar2: %s\n\" "$VAR1" "$VAR2"

- Store a formatted message in a variable (does not work on zsh):
  
  printf -v myvar "This is %s = %d\n\" "a year" 2016

ps

Information about running processes.

- List all running processes:
  
  ps aux

- List all running processes including the full command string:
  
  ps auxww

- Search for a process that matches a string:
  
  ps aux | grep string
**psql**

PostgreSQL command-line client.

- Connect to database. It connects to localhost using default port 5432 with default user as currently logged in user:

  `psql database`

- Connect to database on given server host running on given port with given username, without a password prompt:

  `psql -h host -p port -U username database`

- Connect to database; user will be prompted for password:

  `psql -h host -p port -U username -W database`

- Execute a single SQL query or PostgreSQL command on the given database (useful in shell scripts):

  `psql -c 'query' database`

- Execute commands from a file on the given database:

  `psql database -f file.sql`

**pushd**

Place a directory on a stack so it can be accessed later.

- Switch to directory and push it on the stack:

  `pushd < directory`

- Switch first and second directories on the stack:

  `pushd`

- Rotate stack by making the 5th element the top of the stack:

  `pushd +4`
**pv**

Monitor the progress of data through a pipe.

- Print the contents of the file and display a progress bar:
  ```
  pv file
  ```

- Measure the speed and amount of data flow between pipes (-s is optional):
  ```
  command1 | pv -s expected_amount_of_data_for_eta | command2
  ```

- Filter a file, see both progress and amount of output data:
  ```
  pv -cN in big_text_file | grep pattern | pv -cN out > filtered_file
  ```

- Attach to an already running process and see its file reading progress:
  ```
  pv -d PID
  ```

- Read an erroneous file, skip errors as dd `conv=sync,noerror` would:
  ```
  pv -EE path/to/faulty_media > image.img
  ```

- Stop reading after reading specified amount of data, rate limit to 1K/s:
  ```
  pv -L 1K -S maximum_file_size_to_be_read
  ```

**pwd**

Print name of current/working directory.

- Print the current directory:
  ```
  pwd
  ```

- Print the current directory, and resolve all symlinks (i.e. show the “physical” path):
  ```
  pwd -P
  ```
**pyenv**

Switch between multiple versions of Python easily.

- List all available commands:
  
  `pyenv commands`

- List all Python versions under the `${PYENV_ROOT}/versions` directory:
  
  `pyenv versions`

- Install a Python version under the `${PYENV_ROOT}/versions` directory:
  
  `pyenv install 2.7.10`

- Uninstall a Python version under the `${PYENV_ROOT}/versions` directory:
  
  `pyenv uninstall 2.7.10`

- Set Python version to be used globally in the current machine:
  
  `pyenv global 2.7.10`

- Set Python version to be used in the current directory and all directories below it:
  
  `pyenv local 2.7.10`

**python**

Python language interpreter.

- Call a Python interactive shell (REPL):
  
  `python`

- Execute script in a given Python file:
  
  `python script.py`

- Execute Python language single command:
  
  `python -c command`

- Run library module as a script (terminates option list):
  
  `python -m module arguments`
Execute SQL-like queries on .csv and .tsv files.

- Query .csv file by specifying the delimiter as ',':
  
  `q -d ',' "SELECT * from path/to/file"

- Query .tsv file:
  
  `q -t "SELECT * from path/to/file"

- Query file with header row:
  
  `q -d delimiter -H "SELECT * from path/to/file"

- Read data from stdin; '-' in the query represents the data from stdin:
  
  `output | q "select * from -"

- Join two files (aliased as f1 and f2 in the example) on column c1, a common column:
  
  `q "SELECT * FROM path/to/file f1 JOIN path/to/other_file f2 ON (f1.c1 = f2.c1 )"

- Format output using an output delimiter with an output header line (note: command will output column names based on the input file header or the column aliases overridden in the query):
  
  `q -D delimiter -O "SELECT column as alias from path/to/file"

**qemu-img**

Tool for Quick Emulator Virtual HDD image creation and manipulation.

- Create disk image with a specific size (in gigabytes):
  
  `qemu-img create image_name.img gigabitesG`

- Show information about a disk image:
  
  `qemu-img info image_name.img`

- Increase or decrease image size:
  
  `qemu-img resize image_name.img gigabitesG`

- Dump the allocation state of every sector of the specified disk image:
  
  `qemu-img map image_name.img`
qemu

Generic machine emulator and virtualizer. Supports a large variety of CPU architectures.

- Boot from image emulating i386 architecture:
  ```
  qemu-system-i386 -hda image_name.img
  ```

- Boot from image emulating x64 architecture:
  ```
  qemu-system-x86_64 -hda image_name.img
  ```

- Boot QEMU instance with a live ISO image:
  ```
  qemu-system-i386 -hda image_name.img -cdrom os_image.iso -boot d
  ```

- Specify amount of RAM for instance:
  ```
  qemu-system-i386 -m 256 -hda image_name.img -cdrom os-image.iso -boot d
  ```

- Boot from physical device (e.g. from USB to test bootable medium):
  ```
  qemu-system-i386 -hda /dev/storage_device
  ```

quota

Display users’ disk space usage and allocated limits.

- Show disk quotas for the current user:
  ```
  quota
  ```

- Verbose output (also display quotas on filesystems where no storage is allocated):
  ```
  quota -v
  ```

- Quiet output (only display quotas on filesystems where usage is over quota):
  ```
  quota -q
  ```

- Print quotas for the groups of which the current user is a member:
  ```
  quota -g
  ```

- Show disk quotas for another user (must be superuser to do this):
  ```
  sudo quota -u username
  ```
read

BASH builtin for retrieving data from standard input.

- Store data that you type from the keyboard:
  
  ```bash
  read variable
  ```

- Store each of the next lines you enter as values of an array:
  
  ```bash
  read -a array
  ```

- Enable backspace and GNU readline hotkeys when entering input with read:
  
  ```bash
  read -e variable
  ```

- Specify the number of maximum characters to be read:
  
  ```bash
  read -n character_count variable
  ```

- Use a specific character as a delimiter instead of a new line:
  
  ```bash
  read -d new_delimiter variable
  ```

readlink

Follow symlinks and get symlink information.

- Get the actual file to which the symlink points:
  
  ```bash
  readlink filename
  ```

- Get the absolute path to a file:
  
  ```bash
  readlink -f filename
  ```
**redis-cli**

Opens a connection to a Redis server.

- Connect to the local server:
  
  `redis-cli`

- Connect to a remote server on the default port (6379):
  
  `redis-cli -h host`

- Connect to a remote server specifying a port number:
  
  `redis-cli -h host -p port`

- Specify a password:
  
  `redis-cli -a password`

- Execute Redis command:
  
  `redis-cli redis_command`

**redshift**

Adjust the color temperature of your screen according to your surroundings.

- Turn on Redshift with 5700K temperature during day and 3600K at night:
  
  `redshift -t 5700:3600`

- Turn on Redshift with a manually-specified custom location:
  
  `redshift -l latitude:longitude`

- Turn on Redshift with 70% screen brightness during day and 40% brightness at night:
  
  `redshift -b 0.7:0.4`

- Turn on Redshift with custom gamma levels (between 0 and 1):
  
  `redshift -g red:green:blue`

- Turn on Redshift with a constant unchanging color temperature:
  
  `redshift -0 temperature`
rename

Renames multiple files.

- Rename files using a Perl Common Regular Expression (substitute ‘foo’ with ‘bar’ wherever found):
  
  `rename 's/foo/bar/' \*`

- Dry-run - display which renames would occur without performing them:
  
  `rename -n 's/foo/bar/' \*`

- Force renaming even if the operation would overwrite existing files:
  
  `rename -f 's/foo/bar/' \*`

- Convert filenames to lowercase (use -f in case-insensitive filesystems to prevent “already exists” errors):
  
  `rename 'y/A-Z/a-z/' \*`

- Replace whitespace with underscores:
  
  `rename 's/\s+/_/g' \*`

renice

Alters the scheduling priority/nicenesses of one or more running processes. Niceness values range from -20 (most favorable to the process) to 19 (least favorable to the process).

- Change priority of a running process:
  
  `renice -n niceness_value -p pid`

- Change priority of all processes owned by a user:
  
  `renice -n niceness_value -u user`

- Change priority of all processes that belongs to a group:
  
  `renice -n niceness_value -g group`
rev

Reverse a line of text.

- Reverse the text string “hello”:
  
  ```bash
  echo "hello" | rev
  ```

- Reverse an entire file and print to stdout:
  
  ```bash
  rev file
  ```

ripgrep

A fast command-line search tool.

- Recursively search the current directory for a regex pattern:
  
  ```bash
  rg pattern
  ```

- Search for pattern including all .gitignore and hidden files:
  
  ```bash
  rg -uu pattern
  ```

- Search for a pattern only in a certain file type (e.g., html, css, etc.):
  
  ```bash
  rg -t filetype pattern
  ```

- Search for a pattern only in a subset of directories:
  
  ```bash
  rg pattern set_of_subdirs
  ```

- Search for a pattern in files matching a glob (e.g., README.*):
  
  ```bash
  rg pattern -g glob
  ```
**rm**

Remove files or directories.

- Remove files from arbitrary locations:
  
  `rm path/to/file path/to/another/file`

- Recursively remove a directory and all its subdirectories:
  
  `rm -r path/to/folder`

- Forcibly remove a directory, without prompting for confirmation or showing error messages:
  
  `rm -rf path/to/folder`

- Interactively remove multiple files, with a prompt before every removal:
  
  `rm -i file(s)`

- Remove files in verbose mode, printing a message for each removed file:
  
  `rm -v path/to/folder/*`

**rmdir**

Removes a directory.

- Remove directory, provided it is empty. Use `rm` to remove not empty directories:
  
  `rmdir path/to/directory`

- Remove directories recursively (useful for nested dirs):
  
  `rmdir -p path/to/directory`

**route**

Use `route` cmd to set the route table.

- Display the information of route table:
  
  `route -n`

- Add route rule:
  
  `sudo route add -net ip_address netmask netmask_address gw gw_address`

- Delete route rule:
  
  `sudo route del -net ip_address netmask netmask_address dev gw_address`
rsync

Transfer files either to or from a remote host (not between two remote hosts). Can transfer single files, or multiple files matching a pattern.

- Transfer file from local to remote host:
  \texttt{rsync path/to/file remote_host_name:remote_host_location}

- Transfer file from remote host to local:
  \texttt{rsync remote_host_name:remote_file_location local_file_location}

- Transfer file in archive (to preserve attributes) and compressed (zipped) mode:
  \texttt{rsync -az path/to/file remote_host_name:remote_host_location}

- Transfer a directory and all its children from a remote to local:
  \texttt{rsync -r remote_host_name:remote_folder_location local_folder_location}

- Transfer only updated files from remote host:
  \texttt{rsync -ru remote_host_name:remote_folder_location local_folder_location}

- Transfer file over SSH and show progress:
  \texttt{rsync -e ssh --progress remote_host_name:remote_file local_file}

rtv

Reddit Terminal Viewer. Use arrow keys to navigate. Right and Left to view and return from a submission, respectively.

- Open the front page:
  /front

- Open a subreddit:
  /r/subreddit_name

- Expand/collapse comments:
  [space]

- Open link:
  o

- Login:
  u

- Open the help screen:
  ?
**rustc**

The Rust compiler. Processes, compiles and links Rust language source files.

- Compile a single file:
  
rustc file.rs

- Compile with high optimization:
  
rustc -O file.rs

- Compile with debugging information:
  
rustc -g file.rs

**sails**

Sails.js is a realtime enterprise level MVC framework built on top of Node.js.

- Start Sails:
  
sails lift

- Create new Sails project:
  
sails new projectName

- Generate Sails API:
  
sails generate name

- Generate Sails Controller:
  
sails generate controller name

- Generate Sails Model:
  
sails generate model name
**salt-key**

Invoke salt locally on a salt minion.

- Perform a highstate on this minion:

  salt-call state.highstate

- Perform a highstate dry-run, compute all changes but don’t actually perform them:

  salt-call state.highstate test=true

- Perform a highstate with verbose debugging output:

  salt-call -l debug state.highstate

- List this minion’s grains:

  salt-call grains.items

**salt-key**

Manages salt minion keys on the salt master. Needs to be run on the salt master, likely as root or with sudo.

- List all accepted, unaccepted and rejected minion keys:

  salt-key -L

- Accept a minion key by name:

  salt-key -a MINION_ID

- Reject a minion key by name:

  salt-key -r MINION_ID

- Print fingerprints of all public keys:

  salt-key -F
**salt-run**

Frontend for executing salt-runners on minions.

- Show status of all minions:
  
salt-run manage.status

- Show all minions which are disconnected:
  
salt-run manage.up

**salt**

Execute commands and assert state on remote salt minions.

- List connected minions:
  
salt '*' test.ping

- Execute a highstate on all connected minions:
  
salt '*' state.highstate

- Upgrade packages using the OS package manager (apt, yum, brew) on a subset of minions:
  
salt '*.domain.com' pkg.upgrade

- Execute an arbitrary command on a particular minion:
  
salt 'minion_id' cmd.run "ls "

**samtools**

Tools for handling high-throughput sequencing (genomics) data. Used for reading/writing/editing/indexing/viewing of data in SAM/BAM/CRAM format.

- Convert a SAM input file to BAM stream and save to file:
  
samtools view -S -b input.sam > output.bam

- Take input from stdin (-) and print the SAM header and any reads overlapping a specific region to stdout:
  
other_command | samtools view -h - chromosome:start-end
- Sort file and save to BAM (the output format is automatically determined from the output file's extension):
  ```bash
  samtools sort input -o output.bam
  ```
- Index a sorted BAM file (creates `sorted_input.bam.bai`):
  ```bash
  samtools index sorted_input.bam
  ```
- Print alignment statistics about a file:
  ```bash
  samtools flagstat sorted_input
  ```
- Count alignments to each index (chromosome/contig):
  ```bash
  samtools idxstats sorted_indexed_input
  ```
- Merge multiple files:
  ```bash
  samtools merge output input_1 [input_2...]
  ```
- Split input file according to read groups:
  ```bash
  samtools split merged_input
  ```

**sass**

Converts SCSS or Sass files to CSS.
- Convert a SCSS or Sass file to CSS and print out the result:
  ```bash
  sass inputfile.scss|inputfile.sass
  ```
- Convert a SCSS or Sass file to CSS and save the result to a file:
  ```bash
  sass inputfile.scss|inputfile.sass outputfile.css
  ```
- Watch a SCSS or Sass file for changes and output or update the CSS file with same filename:
  ```bash
  sass --watch inputfile.scss|inputfile.sass
  ```
- Watch a SCSS or Sass file for changes and output or update the CSS file with the given filename:
  ```bash
  sass --watch inputfile.scss|inputfile.sass:outputfile.css
  ```
**scp**

Secure copy. Copy files between hosts using Secure Copy Protocol over SSH.

- Copy a local file to a remote host:
  `scp path/to/local_file remote_host:path/to/remote_file`

- Copy a file from a remote host to a local folder:
  `scp remote_host:path/to/remote_file path/to/local_dir`

- Recursively copy the contents of a directory on a remote host to a local directory:
  `scp -r path/to/local_dir remote_host:path/to/remote_dir`

- Copy a file between two remote hosts transferring through the local host:
  `scp -3 host1:path/to/remote_file host2:path/to/remote_dir`

- Use a specific username when connecting to the remote host:
  `scp path/to/local_file remote_username@remote_host:path/to/remote_dir`

- Use a specific ssh private key for authentication with the remote host:
  `scp -i ~//.ssh/private_key local_file remote_host:/path/remote_file`

**scrapy**

Web-crawling framework.

- Create a project:
  `scrapy startproject project_name`

- Create a spider (in project directory):
  `scrapy genspider spider_name website_domain`

- Edit spider (in project directory):
  `scrapy edit spider_name`

- Run spider (in project directory):
  `scrapy crawl spider_name`

- Fetch a webpage as scrapy sees it and print source in stdout:
  `scrapy fetch url`

- Open a webpage in the default browser as scrapy sees it (disable javascript for extra fidelity):
  `scrapy view url`

- Open scrapy shell for url, which allows interaction with the page source in python shell (or ipython if available):
  `scrapy shell url`
**screen**

Hold a session open on a remote server. Manage multiple windows with a single SSH connection.

- Start a new screen session:
  ```bash
screen
  ```
- Start a new named screen session:
  ```bash
  screen -S session_name
  ```
- Start a new deamon and log the output to screenlog.x:
  ```bash
  screen -dmLS session_name command
  ```
- Show open screen sessions:
  ```bash
  screen -ls
  ```
- Reattach to an open screen:
  ```bash
  screen -r session_name
  ```
- Detach from inside a screen:
  ```bash
  ctrl+A D
  ```
- Kill a detached screen:
  ```bash
  screen -X -S session_name quit
  ```

**screenfetch**

Display system information.

- Start screenfetch:
  ```bash
  screenfetch
  ```
- Take a screenshot (requires 'scrot'):
  ```bash
  screenfetch -s
  ```
- Specify distribution logo:
  ```bash
  screenfetch -A 'distribution_name'
  ```
- Specify distribution logo and text:
  ```bash
  screenfetch -D 'distribution_name'
  ```
- Strip all color:
  ```bash
  screenfetch -N
  ```
**sed**

Run replacements based on regular expressions.

- Replace the first occurrence of a string in a file, and print the result:

  ```
  sed 's/find/replace/' filename
  ```

- Replace all occurrences of an extended regular expression in a file:

  ```
  sed -r 's/regex/replace/g' filename
  ```

- Replace all occurrences of a string in a file, overwriting the file (i.e. in-place):

  ```
  sed -i 's/find/replace/g' filename
  ```

- Replace only on lines matching the line pattern:

  ```
  sed '/line_pattern/s/find/replace/' filename
  ```

- Apply multiple find-replace expressions to a file:

  ```
  sed -e 's/find/replace/' -e 's/find/replace/' filename
  ```

- Replace separator / by any other character not used in the find or replace patterns, e.g., #:

  ```
  sed 's#find#replace#' filename
  ```

**sendmail**

Send email from the command line.

- Send a message with the content of message.txt to the mail folder of local user user_name:

  ```
  sendmail user_name < message.txt
  ```

- Send an email from you@yourdomain.com (assuming your local mail server is configured for this) to test@gmail.com containing the message in message.txt:

  ```
  sendmail -f test@gmail.com you@yourdomain.com < message.txt
  ```

- Send an email from you@yourdomain.com (assuming your local mail server is configured for this) to test@gmail.com containing the file file.zip:

  ```
  sendmail -f test@gmail.com you@yourdomain.com < file.zip
  ```
**seq**

Output a sequence of numbers to stdout.

- Sequence from 1 to 10:
  ```
  seq 10
  ```
- Every 3rd number from 5 to 20:
  ```
  seq 5 3 20
  ```
- Separate the output with a space instead of a newline:
  ```
  seq -s " " 5 3 20
  ```

**sftp**

Secure File Transfer Program. Interactive program to copy files between hosts over SSH. For non-interactive file transfers, see scp or rsync.

- Connect to a remote server and enter an interactive command mode:
  ```
  sftp remote_user@remote_host
  ```
- Connect using an alternate port:
  ```
  sftp -P remote_port remote_user@remote_host
  ```
- Transfer remote file to the local system:
  ```
  get /path/remote_file
  ```
- Transfer local file to the remote system:
  ```
  put /path/local_file
  ```
- Transfer remote folder to the local system recursively (works with put too):
  ```
  get -R /path/remote_folder
  ```
- Get list of files on local machine:
  ```
  ll
  ```
- Get list of files on remote machine:
  ```
  ls
  ```
**sh**

Bourne shell. The standard command language interpreter.

- Start interactive shell:
  
  `sh`

- Execute a command:
  
  `sh -c command`

- Run commands from a file:
  
  `sh file.sh`

- Run commands from STDIN:
  
  `sh -s`

**shopt**

Manage Bash shell options: variables (stored in `$BASHOPTS`) that control behavior specific to the Bash shell. Generic POSIX shell variables (stored in `$SHELLOPTS`) are managed with the `set` command instead.

- List of all settable options and whether they are set:
  
  `shopt`

- Set an option:
  
  `shopt -s option_name`

- Unset an option:
  
  `shopt -u option_name`

- Print a list of all options and their status formatted as runnable `shopt` commands:
  
  `shopt -p`

- Show help for the command:
  
  `help shopt`
shred

Overwrite files to securely delete data.

- Overwrite a file:
  
  shred file

- Overwrite a file, leaving zeroes instead of random data:
  
  shred --zero file

- Overwrite a file 25 times:
  
  shred -n25 file

- Overwrite a file and remove it:
  
  shred --remove file

skicka

Manage your Google Drive.

- Upload a file/folder to Google Drive:
  
  skicka upload path/to/local path/to/remote

- Download a file/folder from Google Drive:
  
  skicka download path/to/remote path/to/local

- List files:
  
  skicka ls path/to/folder

- Show amount of space used by children folders:
  
  skicka du path/to/parent/folder

- Create a folder:
  
  skicka mkdir path/to/folder

- Delete a file:
  
  skicka rm path/to/file
sl

Steam locomotive running through your terminal.
- Let a steam locomotive run through your terminal:
  \texttt{sl}
- The train burns, people scream:
  \texttt{sl -a}
- Let the train fly:
  \texttt{sl -F}

\textbf{slackcat}

Utility for passing files and command output to Slack.
- Post a file to Slack:
  \texttt{slackcat --channel channel\_name path/to/file}
- Post a file to Slack with a custom filename:
  \texttt{slackcat --channel channel\_name --filename=filename path/to/file}
- Pipe command output to Slack as a text snippet:
  \texttt{command | slackcat --channel channel\_name --filename=snippet\_name}
- Stream command output to Slack continuously:
  \texttt{command | slackcat --channel channel\_name --stream}

\textbf{sleep}

Delay for a specified amount of time.
- Delay in seconds:
  \texttt{sleep seconds}
- Delay in minutes:
  \texttt{sleep minutesm}
- Delay in hours:
  \texttt{sleep hoursh}
**socat**

Multipurpose relay (SOcket CAT).

- Listen to a port, wait for an incoming connection and transfer data to STDIO:
  ```
  socat - TCP-LISTEN:8080,fork
  ```

- Create a connection to a host and port, transfer data in STDIO to connected host:
  ```
  socat - TCP4:www.domain.com:80
  ```

- Forward incoming data of a local port to another host and port:
  ```
  socat TCP-LISTEN:80,fork TCP4:www.domain.com:80
  ```

**sort**

Sort lines of text files.

- Sort a file in ascending order:
  ```
  sort filename
  ```

- Sort a file in descending order:
  ```
  sort -r filename
  ```

- Sort a file using numeric rather than alphabetic order:
  ```
  sort -n filename
  ```

- Sort the passwd file by the 3rd field, numerically:
  ```
  sort -t: -k 3n /etc/passwd
  ```

- Sort a file preserving only unique lines:
  ```
  sort -u filename
  ```
**sox**

Sound eXchange: play, record and convert audio files. Audio formats are identified by the extension.

- Merge two audio files into one:
  
  sox -m input_audiofile1 input_audiofile2 output_audiofile

- Trim an audio file to the specified times:
  
  sox input_audiofile output_audiofile trim start end

- Normalize an audio file (adjust volume to the maximum peak level, without clipping):
  
  sox --norm input_audiofile output_audiofile

- Reverse and save an audio file:
  
  sox input_audiofile output_audiofile reverse

- Print statistical data of an audio file:
  
  sox input_audiofile -n stat

- Increase the volume of an audio file by 2x:
  
  sox -v 2.0 input_audiofile output_audiofile

**split**

Split a file into pieces.

- Split a file, each split having 10 lines (except the last split):
  
  split -l 10 filename

- Split a file into 5 files. File is split such that each split has same size (except the last split):
  
  split -n 5 filename

- Split a file with at most 512 bytes of lines in each split:
  
  split -C 512 filename
**srm**

Securely remove files or directories. Overwrites the existing data one or multiple times. Drop in replacement for rm.

- Remove a file after a single-pass overwriting with random data:
  
  ```
  srm -s /path/to/file
  ```

- Remove a file after seven passes of overwriting with random data:
  
  ```
  srm -m /path/to/file
  ```

- Recursively remove a directory and its contents overwriting each file with a single-pass of random data:
  
  ```
  srm -r -s /path/to/folder
  ```

- Prompt before every removal:
  
  ```
  srm -i *
  ```

**ssh-copy-id**

Install your public key in a remote machine’s authorized_keys.

- Copy your keys to the remote machine:
  
  ```
  ssh-copy-id username@remote_host
  ```

- Copy the given public key to the remote:
  
  ```
  ssh-copy-id -i path/to/certificate username@remote_host
  ```

- Copy the given public key to the remote with specific port:
  
  ```
  ssh-copy-id -i path/to/certificate -p port username@remote_host
  ```
**ssh-keygen**

Generate ssh keys user for authentication, password-less logins, and other things.

- Generate a key interactively:

  ```
  ssh-keygen
  ```

- Specify file in which to save the key:

  ```
  ssh-keygen -f ~/.ssh/username
  ```

- Generate a DSA 2048 bit (default) key:

  ```
  ssh-keygen -t dsa
  ```

- Generate an RSA 4096 bit key with your email as a comment:

  ```
  ssh-keygen -t rsa -b 4096 -C "email"
  ```

- Retrieve the key fingerprint from a host (useful for confirming the authenticity of the host when first connecting to it via SSH):

  ```
  ssh-keygen -l -F remote_host
  ```

- Retrieve the fingerprint of a key in MD5 Hex:

  ```
  ssh-keygen -l -E md5 -f ~/.ssh/username
  ```

- Change the password of a key:

  ```
  ssh-keygen -p -f ~/.ssh/username
  ```

**ssh**

Secure Shell is a protocol used to securely log onto remote systems. It can be used for logging or executing commands on a remote server.

- Connect to a remote server:

  ```
  ssh username@remote_host
  ```

- Connect to a remote server with a specific identity (private key):

  ```
  ssh -i path/to/key_file username@remote_host
  ```
- Connect to a remote server using a specific port:

  ssh username@remote_host -p 2222

- Run a command on a remote server:

  ssh remote_host command -with -flags

- SSH tunneling: Dynamic port forwarding (SOCKS proxy on localhost:9999):

  ssh -D 9999 -C username@remote_host

- SSH tunneling: Forward a specific port (localhost:9999 to slashdot.org:80):

  ssh -L 9999:slashdot.org:80 username@remote_host

- Enable the option to forward the authentication information to the remote machine (see man ssh_config for available options):

  ssh -o "ForwardAgent=yes" username@remote_host

**sshfs**

Filesystem client based on ssh.

- Mount remote directory:

  sshfs username@remote_host:remote_directory mountpoint

- Unmount remote directory:

  fusermount -u mountpoint

- Mount remote directory from server with specific port:

  sshfs username@remote_host:remote_directory -p 2222

- Use compression:

  sshfs username@remote_host:remote_directory -C
**st-flash**

Flash binary files to STM32 ARM Cortex microcontrollers.

- Read 4096 bytes from the device starting from 0x8000000:
  
  ```
  st-flash read firmware.bin 0x8000000 4096
  ```

- Write firmware to device starting from 0x8000000:
  
  ```
  st-flash write firmware.bin 0x8000000
  ```

- Erase firmware from device:
  
  ```
  st-flash erase
  ```

**st-info**

Provides information about connected STLink and STM32 devices.

- Display amount of program memory available:
  
  ```
  st-info --flash
  ```

- Display amount of sram memory available:
  
  ```
  st-info --sram
  ```

- Display summarized information of the device:
  
  ```
  st-info --probe
  ```

**st-util**

Run GDB (GNU Debugger) server to interact with STM32 ARM Cortex microcontroller.

- Run GDB server on port 4500:
  
  ```
  st-util -p 4500
  ```

- Connect to GDB server:
  
  ```
  (gdb) target extended-remote localhost:4500
  ```

- Write firmware to device:
  
  ```
  (gdb) load firmware.elf
  ```
strings

Find printable strings in an object file or binary.

- Print all strings in a binary:
  
  strings file

- Limit results to strings at least length characters long:
  
  strings -n length file

- Prefix each result with its offset within the file:
  
  strings -t d file

- Prefix each result with its offset within the file in hexadecimal:
  
  strings -t x file

su

Switch shell to another user.

- Switch to user {{username}} (password required):
  
  su username

- Switch to superuser (admin password required):
  
  su

- Switch to user {{username}} and simulate a full login shell:
  
  su - username

subliminal

Python-based subtitle downloader.

- Download English subtitles for a video:
  
  subliminal download -l en video.ext
**sudo**

Execute a command as another user.

- List of an unreadable directory:
  
  ```
  sudo ls /usr/local/scrt
  ```

- To edit a file as user www:
  
  ```
  sudo -u www vi /var/www/index.html
  ```

- To shutdown the machine:
  
  ```
  sudo shutdown -h +10 "Cya soon!"
  ```

- To repeat the last command as sudo:
  
  ```
  sudo !!
  ```

**sum**

Compute checksums and the number of blocks for a file. A predecessor to the more modern `cksum`.

- Compute a checksum with BSD-compatible algorithm and 1024-byte blocks:
  
  ```
  sum file
  ```

- Compute a checksum with System V-compatible algorithm and 512-byte blocks:
  
  ```
  sum --sysv file
  ```

**supervisorctl**

Supervisor is a client/server system that allows its users to control a number of processes on UNIX-like operating systems. `supervisorctl` is the command-line client piece of the supervisor which provides a shell-like interface.

- Start/stop/restart a process:
  
  ```
  supervisorctl start|stop|restart process_name
  ```

- Start/stop/restart all processes in a group:
supervisorctl start|stop|restart group_name:* 
- Show last 100 **bytes** of process stderr:
  supervisorctl tail -100 process_name stderr 
- Keep displaying stdout of a process:
  supervisorctl tail -f process_name stdout 
- Reload process config file to add/remove processes as necessary:
  supervisorctl update 

**supervisord**

Supervisor is a client/server system for controlling some processes on UNIX-like operating systems. Supervisord is the server part of supervisor; it is primarily managed via a configuration file.
- Start supervisord with specified configuration file:
  supervisord -c path/to/file 
- Run supervisord in the foreground:
  supervisord -n 

**svn**

Subversion command line client tool.
- Check out a working copy from a repository:
  svn co url/to/repository  
- Bring changes from the repository into the working copy:
  svn up 
- Put files and directories under version control, scheduling them for addition to repository. They will be added in next commit:
  svn add PATH... 
- Send changes from your working copy to the repository:
  svn ci -m commit log message [PATH...] 
- Show detailed help:
  svn help
**tabula**

Extract tables from PDF files.

- Extract all tables from a PDF to a CSV file:
  ```
tabula -o file.csv file.pdf
  ```

- Extract all tables from a PDF to a JSON file:
  ```
tabula --format JSON -o file.json file.pdf
  ```

- Extract tables from pages 1, 2, 3, and 6 of a PDF:
  ```
tabula --pages 1-3,6 file.pdf
  ```

- Extract tables from page 1 of a PDF, guessing which portion of the page to examine:
  ```
tabula --guess --pages 1 file.pdf
  ```

- Extract all tables from a PDF, using ruling lines to determine cell boundaries:
  ```
tabula --spreadsheet file.pdf
  ```

- Extract all tables from a PDF, using blank space to determine cell boundaries:
  ```
tabula --no-spreadsheet file.pdf
  ```

**tac**

Print and concatenate files in reverse.

- Print the contents of file1 reversed to the standard output:
  ```
tac file1
  ```

- Concatenate several files reversed into the target file:
  ```
tac file1 file2 > target_file
  ```
**tail**

Display the last part of a file.

- Show last ‘num’ lines in file:
  `tail -n num file`
- Show all file since line ‘num’:
  `tail -n +num file`
- Show last ‘num’ bytes in file:
  `tail -c num file`
- Keep reading file until ctrl-c:
  `tail -f file`

**tar**

Archiving utility. Often combined with a compression method, such as gzip or bzip.

- Create an archive from files:
  `tar cf target.tar file1 file2 file3`
- Create a gzipped archive:
  `tar czf target.tar.gz file1 file2 file3`
- Extract an archive in a target folder:
  `tar xf source.tar -C folder`
- Extract a gzipped archive in the current directory:
  `tar xzf source.tar.gz`
- Extract a bzipped archive in the current directory:
  `tar xjf source.tar.bz2`
- Create a compressed archive, using archive suffix to determine the compression program:
  `tar caf target.tar.xz file1 file2 file3`
- List the contents of a tar file:
  `tar tvf source.tar`
task

TODO list manager.
- Add new task:
  task add thing_to_do
- List tasks:
  task list
- Mark task as completed:
  task task_id done
- Modify task:
  task task_id modify new_thing_to_do
- Delete task:
  task task_id delete

tcpdump

Dump traffic on a network.
- Capture the traffic of a specific interface:
  tcpdump -i eth0
- Capture all TCP traffic showing contents (ASCII) in console:
  tcpdump -A tcp
- Capture the traffic from or to a host:
  tcpdump host www.example.com
- Capture the traffic from a specific interface, source, destination and destination port:
  tcpdump -i eth0 src 192.168.1.1 and dst 192.168.1.2 and dst port 80
- Capture the traffic of a network:
  tcpdump net 192.168.1.0/24
- Capture all traffic except traffic over port 22 and save to a dump file:
  tcpdump -w dumpfile.pcap not port 22
**tee**

Read from standard input and write to standard output and files.

- Copy standard input to each FILE, and also to standard output:
  
  ```
  echo "example" | tee FILE
  ```

- Append to the given FILEs, do not overwrite:
  
  ```
  echo "example" | tee -a FILE
  ```

**telnet**

Telnet is used to connect to a specified port of a host.

- Telnet to a certain port:
  
  ```
  telnet ip_address port
  ```

- To exit a telnet session:
  
  ```
  quit
  ```

- Default escape character:
  
  ```
  CTRL + ]
  ```

- Specify an escape character (x is the escape character):
  
  ```
  telnet -e x ip_address port
  ```

**tesseract**

OCR (Optical Character Recognition) engine.

- Recognize text in an image and save it to output.txt. The file extension MUST not be mentioned:
  
  ```
  tesseract image.png output
  ```

- Specify a custom language (default is English) with an ISO 639-2 code (e.g. deu = Deutsch = German):
  
  ```
  tesseract -l deu image.png output
  ```
- List the ISO 639-2 codes of available languages:

  `tesseract --list-langs`

- Specify a custom page segmentation mode (default is 3):

  `tesseract -psm 0_to_10 image.png out`

- List page segmentation modes and their descriptions:

  `tesseract --help-psm`

**test**

Evaluate condition. If it is true, returns 0 exit status, otherwise returns 1.

- Test if given variable is equal to given string:

  `test $MY_VAR == '/bin/zsh'`

- Test if given variable is empty:

  `test -z $GIT_BRANCH`

- Test if file exists:

  `test -e filename`

- Test if directory not exists:

  `test ! -d path/to/directory`

- If-else statement:

  `test condition && echo "true" || echo "false"`

**time**

See how long a command takes.

- Time "ls":

  `time ls`
**tldr**

Simplified man pages.

- Get typical usages of a command (hint: this is how you got here!):
  
  tldr command

- Update the local cache of tldr pages:
  
  tldr --update

**tldrl**

Lint and format TLDR pages.

- Lint all pages:
  
  tldrl pages_directory

- Format a specific page to stdout:
  
  tldrl -f page.md

- Format all pages in place:
  
  tldrl -fi pages_directory

**tmux**

Multiplex several virtual consoles.

- Start a new tmux session:
  
  tmux

- Start a new named tmux session:
  
  tmux new -s name

- List sessions:
  
  tmux ls

- Attach to a session:
tmux a
- Attach to a named session:
  tmux a -t name
- Detach from session:
  ctrl+b d
- Kill session:
  tmux kill-session -t name

touch
Change a file access and modification times (atime, mtime).
- Create a new empty file(s) or change the times for existing file(s) to current time:
  touch filename
- Set the times on a file to a specific date and time:
  touch -t YYYYMMDDHHMM.SS filename
- Use the times from a file to set the times on a second file:
  touch -r filename filename2

tput
View and modify terminal settings and capabilities.
- Move the cursor to a screen location:
  tput cup y_coordinate x_coordinate
- Set foreground (af) or background (ab) color:
  tput setaf|setab ansi_color_code
- Show number of columns, lines, or colors:
  tput cols|lines|colors
- Ring the terminal bell:
  tput bel
- Reset all terminal attributes:
  tput sgr0
**tr**

Translate characters - run replacements based on single characters and character sets.

- Replace all occurrences of a character in a file, and print the result:
  
  `tr find_characters replace_characters < filename`

- Map each character of the first set to the corresponding character of the second set:
  
  `tr 'abcd' 'jkmn' < filename`

- Delete all occurrences of the specified set of characters from the input:
  
  `tr -d 'input_characters'`

- Compress a series of identical characters to a single character:
  
  `tr -s '\n'`

- Translate the contents of the file to upper-case and print result:
  
  `tr "[:lower:]" "[:upper:]" < filename`

- Strip out non-printable characters from the file and print result:
  
  `tr -cd "[:print:]" < filename`

**traceroute**

Print the route packets trace to network host.

- Traceroute to a host:
  
  `traceroute host`

- Disable IP address and host name mapping:
  
  `traceroute -n host`

- Specify wait time for response:
  
  `traceroute -w 0.5 host`

- Specify number of queries per hop:
  
  `traceroute -q 5 host`

- Specify size in bytes of probing packet:
  
  `traceroute host 42`
**transcode**

Transcode video and audio codecs, and convert between media formats.

- Create stabilisation file to be able to remove camera shakes:
  
  ```bash
  transcode -J stabilize -i input_file
  ```

- Remove camera shakes after creating stabilisation file, transform video using xvid:
  
  ```bash
  transcode -J transform -i input_file -y xvid -o output_file
  ```

- Resize the video to 640x480 pixels and convert to MPEG4 codec using xvid:
  
  ```bash
  transcode -Z 640x480 -i input_file -y xvid -o output_file
  ```

**tty**

Returns terminal name.

- Print the file name of this terminal:
  
  ```bash
  tty
  ```

**ufraw-batch**

Convert RAW files from cameras into standard image files.

- Simply convert RAW files to jpg:
  
  ```bash
  ufraw-batch --out-type=jpg input_file(s)
  ```

- Simply convert RAW files to png:
  
  ```bash
  ufraw-batch --out-type=png input_file(s)
  ```

- Extract the preview image from the raw file:
  
  ```bash
  ufraw-batch --embedded-image input_file(s)
  ```

- Save the file with size up to the given maximums MAX1 and MAX2:
  
  ```bash
  ufraw-batch --size=MAX1,MAX2 input_file(s)
  ```
**umount**

Revokes access to an entire filesystem mounted to a directory. A filesystem cannot be unmounted when it is busy.

- Unmount a filesystem:
  
  ```
  umount path/to/device_file
  ```

- OR:

  ```
  umount path/to/mounted_directory
  ```

- Unmount all mounted filesystems (dangerous!):

  ```
  umount -a
  ```

**unar**

Extract contents from archive files.

- Extract an archive to the current directory:

  ```
  unar archive
  ```

- Extract an archive to the specified directory:

  ```
  unar -o path/to/directory archive
  ```

- Force overwrite if files to be unpacked already exist:

  ```
  unar -f archive
  ```

- Force rename if files to be unpacked already exist:

  ```
  unar -r archive
  ```

- Force skip if files to be unpacked already exist:

  ```
  unar -s archive
  ```
uniq

Output the unique lines from the given input or file. Since it does not detect repeated lines unless they are adjacent, we need to sort them first.

- Display each line once:
  ```bash
  sort file | uniq
  ```

- Display only unique lines:
  ```bash
  sort file | uniq -u
  ```

- Display only duplicate lines:
  ```bash
  sort file | uniq -d
  ```

- Display number of occurrences of each line along with that line:
  ```bash
  sort file | uniq -c
  ```

- Display number of occurrences of each line, sorted by the most frequent:
  ```bash
  sort file | uniq -c | sort -nr
  ```

unrar

Extract RAR archives.

- Extract files with original directory structure:
  ```bash
  unrar x compressed.rar
  ```

- Extract files into current directory, losing directory structure in the archive:
  ```bash
  unrar e compressed.rar
  ```

- Test integrity of each file inside the archive file:
  ```bash
  unrar t compressed.rar
  ```

- List files inside the archive file without decompressing it:
  ```bash
  unrar l compressed.rar
  ```
unzip

Extract compressed files in a ZIP archive.

- Extract zip file(s) (for multiple files, separate file paths by spaces):
  
  `unzip file(s)`

- Extract zip files(s) to given path:
  
  `unzip compressed_file(s) -d /path/to/put/extracted_file(s)`

- List the contents of a zip file without extracting:
  
  `unzip -l file`

uptime

Tell how long the system has been running and other information.

- Print current time, uptime, number of logged-in users and other information:
  
  `uptime`

vagrant

Manage lightweight, reproducible, and portable development environments.

- Create Vagrantfile in current folder with the base Vagrant box:
  
  `vagrant init`

- Create Vagrantfile with the Ubuntu 14.04 (Trusty Tahr) box from HashiCorp Atlas:
  
  `vagrant init ubuntu/trusty32`

- Start and provision the vagrant environment:
  
  `vagrant up`

- Suspend the machine:
  
  `vagrant suspend`

- Connect to machine via SSH:
  
  `vagrant ssh`
vim

Vi IMproved, a programmer’s text editor, providing several modes for different kinds of text manipulation. Pressing i enters edit mode. <Esc> goes back to normal mode, which doesn’t allow regular text insertion.

- Open a file:
  ```
  vim file
  ```
- Enter text editing mode (insert mode):
  ```
  <Esc>i
  ```
- Copy ("yank") or cut ("delete") the current line (paste it with P):
  ```
  <Esc>yy|dd
  ```
- Undo the last operation:
  ```
  <Esc>u
  ```
- Search for a pattern in the file (press n/N to go to next/previous match):
  ```
  <Esc>/search_pattern<Enter>
  ```
- Perform a regex substitution in the whole file (from the start, 1, to the end, $):
  ```
  <Esc>:1,$/pattern/replacement/g<Enter>
  ```
- Save (write) the file, and quit:
  ```
  <Esc>:wq<Enter>
  ```
- Quit without saving:
  ```
  <Esc>:q!<Enter>
  ```

vimtutor

Vim tutor, teaching the basic vim commands.

- Launch the vim tutor using the given language (en, fr, de, ...):
  ```
  vimtutor language
  ```
- Exit the tutor:
  ```
  <Esc> :q <Enter>
  ```
**virtualenv**

Create virtual isolated Python environments.

- Create a new environment:
  ```
  virtualenv path/to/venv
  ```

- Start (select) the environment:
  ```
  source path/to/venv/bin/activate
  ```

- Stop the environment:
  ```
  deactivate
  ```

**visudo**

Safely edit the sudoers file.

- Edit sudoers file:
  ```
  sudo visudo
  ```

- Check sudoers file for errors:
  ```
  sudo visudo -c
  ```

**vue-cli**

Simple CLI for scaffolding Vue.js projects. Official templates include: webpack, webpack-simple, browserify, browserify-simple, simple.

- Create a new vue project:
  ```
  vue init template project_name
  ```

- Create a new project with a local template:
  ```
  vue init path/to/template_folder project_name
  ```

- Create project using template on GitHub:
  ```
  vue init username/repo project_name
  ```
w

Show who is logged on and what they are doing. Print user login, TTY, remote host, login time, idle time, current process.
- Show logged-in users info:
  w
- Show logged-in users info without a header:
  w -h

w3m

A text-based web browser.
- Open an URL:
  w3m http://example.com
- Quit w3m:
  'q' then 'y'

wait

Wait for a process to complete before proceeding.
- Wait for a process to finish given its process ID (PID) and return its exit status:
  wait pid
- Wait for all processes known to the invoking shell to finish:
  wait

watch

Execute a program periodically, showing output fullscreen.
- Repeatedly run a command and show the result:
  watch command
- Re-run a command every 60 seconds:
  watch -n 60 command
- Monitor the contents of a directory, highlighting differences as they appear:
  watch -d ls -l
**wc**

Count words, bytes, or lines.

- Count lines in file:
  
  `wc -l file`

- Count words in file:
  
  `wc -w file`

- Count characters (bytes) in file:
  
  `wc -c file`

- Count characters in file (taking multi-byte character sets into account):
  
  `wc -m file`

**webpack**

Bundle a web project’s js files and other assets into a single output file.

- Create a single output file from an entry point file:
  
  `webpack app.js bundle.js`

- Load css files too from the js file (this uses the css loader for .css files):
  
  `webpack app.js bundle.js --module-bind 'css=css'`

- Pass a config file (with eg. the entry script and the output filename) and show compilation progress:
  
  `webpack --config webpack.config.js --progress`

- Automatically recompile on changes to project files:
  
  `webpack --watch app.js bundle.js`
wget

Download files from the Web. Supports HTTP, HTTPS, and FTP.

- Download the contents of an URL to a file:
  `wget -O filename url`

- Limit download speed:
  `wget --limit-rate=200k url`

- Continue an incomplete download:
  `wget -c url`

- Download a full website:
  `wget --mirror -p --convert-links -P target_folder url`

- FTP download with username and password:
  `wget --ftp-user=username --ftp-password=password url`

- Retry a given number of times if the download doesn't succeed at first:
  `wget -t number_of_retries url`

which

Locate the a program in the user’s path.

- Search the PATH environment variable and display the location of any matching executables:
  `which executable`

- If there are multiple executables which match, display all:
  `which -a executable`

while

Simple shell loop.

- Read stdin and perform an action on every line:
  `while read line; do echo "$line"; done`

- Execute a command forever once every second:
  `while ::; do command; sleep 1; done`
**who**

Display who is logged in and related data (processes, boot time).

- Display the username, line, and time of all currently logged-in sessions:
  
  `who`

- Display information only for the current terminal session:
  
  `who am i`

- Display all available information:
  
  `who -a`

- Display all available information with table headers:
  
  `who -a -H`

**whoami**

Show the username of the current user.

- Display currently logged user name:
  
  `whoami`

**x_x**

View Excel and CSV files from the command-line.

- View an XLSX or CSV file:
  
  `x_x file.xlsx|file.csv`

- View an XLSX or CSV file, using the first row as table headers:
  
  `x_x -h 0 file.xlsx|file.csv`

- View a CSV file with unconventional delimiters:
  
  `x_x --delimiter=';'; --quotechar='|' file.csv`
**xargs**

Execute a command with piped arguments coming from another command, a file, etc. The input is treated as a single block of text and split into separate arguments on spaces, tabs, newlines and end-of-file.

- Main usage pattern:

  `arguments_source | xargs command`

- Delete all files with a `.backup` extension:

  `find . -name '*.backup' | xargs rm -v`

- Convert newlines in the input into NUL (\0) characters, and split on those only (useful if the input to xargs contains spaces):

  `arguments_source | tr '\n' '\0' | xargs -0 command`

- Execute the command once for each input line, replacing any occurrences of the placeholder (here marked as `_`) with the input line:

  `arguments_source | xargs -I _ command _ optional_extra_arguments`

**xcv**

Cut, copy, and paste in the command-line.

- Cut a file:

  `xcv x input_file`

- Copy a file:

  `xcv c input_file`

- Paste a file:

  `xcv v output_file`

- List files available for pasting:

  `xcv l`
**xz**

Compress or decompress .xz and .lzma files.

- Compress a file:
  
  \texttt{xz file}

- Decompress a file:

  \texttt{xz -d file.xz}

- Decompress a file and write to stdout:

  \texttt{xz -dc file.xz}

- Compress a file, but don’t delete the original:

  \texttt{xz -k file}

- Compress a file using the fastest compression:

  \texttt{xz -0 file}

- Compress a file using the best compression:

  \texttt{xz -9 file}

**yarn**

JavaScript and Node.js package manager alternative.

- Install a module globally:

  \texttt{yarn global add module_name}

- Install all dependencies referenced in the package.json file:

  \texttt{yarn}

- Install a module and save it as a dependency to the package.json file (add \texttt{--dev} to save as a dev dependency):

  \texttt{yarn add module_name@version}

- Uninstall a module and remove it from the package.json file:

  \texttt{yarn remove module_name}

- Interactively create a package.json file:

  \texttt{yarn init}

- Identify whether a module is a dependency and list other modules that depend upon it:

  \texttt{yarn why module_name}
yes

Output something repeatedly.

- Repeatedly output "message":
  yes message
- Repeatedly output "y":
  yes

youtube-dl

Download videos from YouTube and other websites.

- Download a video or playlist:
  youtube-dl https://www.youtube.com/watch?v=oHg5SJYRHA0
- Download the audio from a video and convert it to an MP3:
  youtube-dl -x --audio-format mp3 url
- Download video(s) as MP4 files with custom filenames:
  youtube-dl --format mp4 --output "%(title) by %(uploader) on %(upload_date) in %(playlist).%(ext)" url
- Download a video and save its description, metadata, annotations, subtitles, and thumbnail:
  youtube-dl --write-description --write-info-json --write-annotations --write-sub --write-thumbnail url
- From a playlist, download all "Let's Play" videos that aren't marked "NSFW" or age-restricted for 7 year-olds:
  youtube-dl --match-title "let's play" --age-limit 7 --reject-title "nsfw" playlist_url

zbarimg

Scan and decode barcodes from image file(s).

- Process an image file:
  zbarimg image_file
**zcat**

Print data from gzip compressed files.

- Print the uncompressed contents of a gzipped file to the standard output:
  
  ```
  zcat file.txt.gz
  ```

**zdb**

ZFS debugger.

- Show detailed configuration of all mounted ZFS zpools:
  
  ```
  zdb
  ```

- Show detailed configuration for a specific ZFS pool:
  
  ```
  zdb -C poolname
  ```

- Show statistics about number, size and deduplication of blocks:
  
  ```
  zdb -b poolname
  ```

**zfs**

Manage ZFS filesystems.

- List all available zfs filesystems:
  
  ```
  zfs list
  ```

- Create a new ZFS filesystem:
  
  ```
  zfs create pool_name/filesystem_name
  ```

- Delete a ZFS filesystem:
  
  ```
  zfs destroy pool_name/filesystem_name
  ```

- Create a Snapshot of a ZFS filesystem:
  
  ```
  zfs snapshot pool_name/filesystem_name@snapshot_name
  ```

- Enable compression on a filesystem:
  
  ```
  zfs set compression=on pool_name/filesystem_name
  ```

- Change mountpoint for a filesystem:
  
  ```
  zfs set mountpoint=/my/mount/path pool_name/filesystem_name
  ```
zip

Package and compress (archive) files into zip file.

- Package and compress a directory and its contents, [r]ecursively:
  
  `zip -r compressed.zip /path/to/dir`

- E[x]clude unwanted files from being added to the compressed archive:
  
  `zip -r compressed.zip path/to/dir -x *.git* \*node_modules\* ...`

- Package and compress multiple directories and files:
  
  `zip -r compressed.zip /path/to/dir1 /path/to/dir2 /path/to/file`

- Add files to an existing zip file:
  
  `zip compressed.zip path/to/file`

- Delete files from an existing zip file:
  
  `zip -d compressed.zip "foo/*.tmp"

zless

View compressed files.

- Page through a compressed archive with less:
  
  `zless file.txt.gz`

zpool

Manage ZFS pools.

- Show the configuration and status of all ZFS zpools:
  
  `zpool status`

- Check a ZFS pool for errors (verifies the checksum of EVERY block). Very CPU and disk intensive:
  
  `zpool scrub pool_name`
- List zpools available for import:
  `zpool import`

- Import a zpool:
  `zpool import pool_name`

- Export a zpool (unmount all filesystems):
  `zpool export pool_name`

- Show the history of all pool operations:
  `zpool history pool_name`

- Create a mirrored pool:
  `zpool create pool_name mirror disk1 disk2 mirror disk3 disk4`

**zsh**

Z SHell. bash and sh-compatible command line interpreter.

- Start interactive command line interpreter:
  `zsh`

- Execute command passed as parameter:
  `zsh -c command`

- Run commands from file (script):
  `zsh file`

- Run commands from file and print them as they are executed:
  `zsh -x file`
2 LINUX

adduser

User addition utility.
- Create a new user with a default home directory and prompts the user to set a password:
  adduser name
- Create a new user without a home directory:
  adduser --no-create-home name
- Create a new user with a home directory at the specified path:
  adduser --home path/to/home name
- Create a new user with the specified shell set as the login shell:
  adduser --shell path/to/shell name
- Create a new user belonging to the specified group:
  adduser --ingroup group name

apt-cache

Debian and Ubuntu package query tool.
- Search for a package in your current sources:
  apt-cache search query
- Show information about a package:
  apt-cache show package
- Show whether a package is installed and up to date:
  apt-cache policy package
- Show dependencies for a package:
  apt-cache depends package
- Show packages that depend on a particular package:
  apt-cache rdepends package
**apt-get**

Debian and Ubuntu package management utility.

- Synchronize list of packages and versions available. This should be run first, before running subsequent apt-get commands:
  
  apt-get update

- Install a new package:

  apt-get install package

- Remove a package:

  apt-get remove package

- Upgrade installed packages to newest available versions:

  apt-get upgrade

- Remove no longer needed packages:

  apt-get autoremove

- Upgrade installed packages (like “upgrade”), but remove obsolete packages and install additional packages to meet new dependencies:

  apt-get dist-upgrade

**apt-key**

Key management utility for the APT Package Manager on Debian and Ubuntu.

- List trusted keys:

  apt-key list

- Add a key to the trusted keystore:

  apt-key add public_key_file.asc

- Delete a key from the trusted keystore:

  apt-key del key_id

- Add a remote key to the trusted keystore:

  wget -qO - https://host.tld/filename.key | apt-key add -

- Add a key from keys server with only key id:

  apt-key adv --keyserver pgp.mit.edu --recv KEYID
**apt**

Package management utility for Debian based distributions.

- Update list of packages and versions available. This should be run before running further apt commands:
  
  apt update

- Search for packages:
  
  apt search package

- Install a new package:
  
  apt install package

- Remove a package (using “purge” instead also removes its configuration files):
  
  apt remove package

- Upgrade installed packages to the newest available versions:
  
  apt upgrade

- Remove no longer needed packages:
  
  apt autoremove

- Upgrade installed packages and remove no longer needed packages:
  
  apt full-upgrade

**aptitude**

Debian and Ubuntu package management utility.

- Synchronize list of packages and versions available. This should be run first, before running subsequent aptitude commands:
  
  aptitude update

- Install a new package and its dependencies:
  
  aptitude install package
- Search for a package:
  
  `aptitude search package`

- Remove a package and all packages depending on it:
  
  `aptitude remove package`

- Do an `aptitude remove package` and remove all config files:
  
  `aptitude purge package`

- Upgrade installed packages to newest available versions:
  
  `aptitude upgrade`

- Upgrade installed packages (like `aptitude upgrade`) including removing obsolete packages and installing additional packages to meet new package dependencies:
  
  `aptitude full-upgrade`

**archey**

Simple tool for stylishly displaying system information.

- Show system information:
  
  `archey`

**at**

Executes commands at a specified time.

- Open an at prompt to create a new set of scheduled commands, press Ctrl+D to save and exit:
  
  `at hh:mm:ss`

- Execute the commands and email the result using a local mailing program such as `sendmail`:
  
  `at hh:mm:ss -m`

- Execute a script at the given time:
  
  `at hh:mm:ss -f path/to/file`
**beep**

A utility to beep the PC speaker.

- Play a beep:
  `beep`

- Play a beep that repeats:
  `beep -r repetitions`

- Play a beep at a specified frequency (Hz) and duration (milliseconds):
  `beep -f frequency -l duration`

- Play each new frequency and duration as a distinct beep:
  `beep -f frequency -l duration -n -f frequency} -l duration`

- Play the C major scale:
  `beep -f 262 -n -f 294 -n -f 330 -n -f 349 -n -f 392 -n -f 440 -n -f 494 -n -f 523`

**brew**

The Homebrew package manager for Linux.

- Search formula:
  `brew search text`

- Install formula:
  `brew install formula`

- List all installed formulae:
  `brew list`

- Get latest version of installed formula (passing no formula updates all installed formulae):
  `brew upgrade formula`

- Update brew:
  `brew update`

- Display information about formula, which contains formula version, installed path, dependencies, etc.:
  `brew info formula`

- Check your system for potential problems:
  `brew doctor`
**bzip2**

A block-sorting file compressor.

- Compress file:
  
  bzip2 path/to/file_to_compress

- Decompress file:
  
  bzip2 -d path/to/compressed_file.bz2

- Decompress to console:
  
  bzip2 -dc path/to/compressed_file.bz2

**chattr**

Change attributes of files or folders.

- Make a file or folder immutable to changes and deletion, even by superuser:
  
  chattr +i path

- Make a file or folder mutable:
  
  chattr -i path

- Recursively make an entire folder and contents immutable:
  
  chattr -R +i folder

**chroot**

Run command or interactive shell with special root directory.

- Run command as new root directory:
  
  chroot /path/to/new/root command

- Specify user and group (ID or name) to use:
  
  chroot --userspec=user:group
**cmus**

Commandline Music Player. Use arrow keys to navigate, <enter/return> to select, and numbers 1-8 switch between different views.

- Open cmus from specified directory:
  ```
  cmus path/to/directory
  ```
- Add file/directory to library:
  ```
  :add path/to/file_or_directory
  ```
- Pause/unpause current song:
  ```
  c
  ```
- Toggle shuffle mode on/off:
  ```
  s
  ```
- Quit cmus:
  ```
  q
  ```

**compose**

An alias to a run-mailcap's action compose. Originally run-mailcap is used to mime-type/file.

- Compose action can be used to compose any existing file or new on default mailcap edit tool:
  ```
  compose filename
  ```
- With run-mailcap:
  ```
  run-mailcap --action=compose filename
  ```

**cryptsetup**

Manage plain dm-crypt and LUKS (Linux Unified Key Setup) encrypted volumes.

- Initialize a LUKS volume (overwrites all data on the partition):
  ```
  cryptsetup luksFormat /dev/sda1
  ```
- Open a LUKS volume and create a decrypted mapping at /dev/mapper/{{target}}:
  ```
  cryptsetup luksOpen /dev/sda1 target
  ```
- Remove an existing mapping:
  ```
  cryptsetup luksClose target
  ```
**date**

Set or display the system date.

- Display the current date using the default locale’s format:
  
  ```
  date +"%c"
  ```

- Display the current date in UTC and ISO 8601 format:
  
  ```
  date -u +"%Y-%m-%dT%H:%M:%SZ"
  ```

- Display the current date as a Unix timestamp (seconds since the Unix epoch):
  
  ```
  date +%s
  ```

- Display a specific date (represented as a Unix timestamp) using the default format:
  
  ```
  date -d @1473305798
  ```

**dd**

Convert and copy a file.

- Make a bootable usb drive from an isohybrid file (such like archlinux-xxx.iso) and show the progress:
  
  ```
  dd if=file.iso of=/dev/usb_drive status=progress
  ```

- Clone a drive to another drive with 4MB block, ignore error and show progress:
  
  ```
  dd if=/dev/source_drive of=/dev/dest_drive bs=4M conv=noerror status=progress
  ```

- Generate a file of 100 random bytes by using kernel random driver:
  
  ```
  dd if=/dev/urandom of=random_file bs=100 count=1
  ```

- Benchmark the write performance of a disk:
  
  ```
  dd if=/dev/zero of=file_1GB bs=1024 count=1000000
  ```
**dnf**

Package management utility for RHEL, Fedora, and CentOS (replaces yum).

- Synchronize list of packages and versions available. This should be run first, before running subsequent dnf commands:

  dnf update

- Install a new package:

  dnf install package

- Install a new package and assume yes to all questions:

  dnf -y install package

- Remove a package:

  dnf remove package

- Upgrade installed packages to newest available versions:

  dnf upgrade

**dpkg-query**

A tool that shows information about installed packages.

- List all installed packages:

  dpkg-query -l

- List installed packages matching a pattern:

  dpkg-query -l 'pattern'

- List all files installed by a package:

  dpkg-query -L package_name

- Show information about a package:

  dpkg-query -s package_name
dpkg

Debian package manager.
- Install a package:
  `dpkg -i /path/to/file`
- Remove a package:
  `dpkg -r package_name`
- List installed packages:
  `dpkg -l pattern`
- List package contents:
  `dpkg -L package_name`
- Find out which package owns a file:
  `dpkg -S file_name`

du

Disk usage: estimate and summarize file and folder space usage.
- List the sizes of a folder and any subfolders, in the given unit (B/KB/MB):
  `du -b|k|m path/to/folder`
- List the sizes of a folder and any subfolders, in human-readable form (i.e. auto-selecting the appropriate unit for each size):
  `du -h path/to/folder`
- Show the size of a single folder, in human readable units:
  `du -sh path/to/folder`
- List the human-readable sizes of a folder and of all the files and folders within it:
  `du -ah path/to/folder`
- List the human-readable sizes of a folder and any subfolders, up to N levels deep:
  `du -h --max-depth=N path/to/folder`
- List the human-readable size of all .jpg files in subfolders of the current folder, and show a cumulative total at the end:
  `du -ch *//*.jpg`
**edit**

An alias to a run-mailcap's action edit. Originally run-mailcap is used to process/edit mime-type/file.

- Edit action can be used to view any file on default mailcap explorer:
  
edit filename

- With run-mailcap:
  
run-mailcap --action=edit filename

**edquota**

Edit quotas for a user or group. By default it operates on all file systems with quotas. Quota information is stored permanently in the quota.user and quota.group files in the root of the filesystem.

- Edit quota of the current user:
  
edquota --user $(whoami)

- Edit quota of a specific user:
  
sudo edquota --user username

- Edit quota for a group:
  
sudo edquota --group group

- Restrict operations to a given filesystem (by default edquota operates on all filesystems with quotas):
  
sudo edquota --file-system filesystem

- Edit the default grace period:
  
sudo edquota -t

- Duplicate a quota to other users:
  
sudo edquota -p reference_user destination_user1 destination_user2
emerge

Gentoo Linux package manager utility.
- Synchronize all packages:
  `emerge --sync`
- Update all packages, including dependencies:
  `emerge -uDNav @world`
- Resume a failed update, skipping the failing package:
  `emerge --resume --skipfirst`
- Install a new package, with confirmation:
  `emerge -av package_name`
- Remove a package, with confirmation:
  `emerge -Cav package_name`
- Remove orphaned packages (that were installed only as dependencies):
  `emerge -avc`
- Search the package database for a keyword:
  `emerge -S keyword`

equery

View information about Portage packages.
- List all installed packages:
  `equery list '*'`
- Search for installed packages in the Portage tree and in overlays:
  `equery list -po package_name`
- List all packages that depend on a given package:
  `equery depends package_name`
- List all packages that a given package depends on:
  `equery depgraph package_name`
- List all files installed by a package:
  `equery files --tree package_name`
**expand**

Convert tabs to spaces.

- Convert tabs in each file to spaces, writing to standard output:

  ```latex
  expand file
  ```

- Convert tabs to spaces, reading from standard input:

  ```latex
  expand
  ```

- Do not convert tabs after non-blanks:

  ```latex
  expand -i file
  ```

- Have tabs a certain number of characters apart, not 8:

  ```latex
  expand -t=number file
  ```

- Use comma separated list of explicit tab positions:

  ```latex
  expand -t=list
  ```

**expr**

Evaluate expressions and manipulate strings.

- Get string length:

  ```latex
  expr length string
  ```

- Evaluate logical or math expression with an operator (‘+, ‘-, ‘*, ‘&,’ |’, etc.). Special symbols should be escaped:

  ```latex
  expr first_argument operator second_argument
  ```

- Get position of the first character in 'string' that matches 'substring':

  ```latex
  echo $(expr index string substring)
  ```

- Extract part of the string:

  ```latex
  echo $(expr substr string position_to_start number_of_characters)
  ```

- Extract part of the string which matches a regular expression:

  ```latex
  echo $(expr string : '\(regular_expression\)')
  ```
**fc-list**

List available fonts installed on the system.

- Return a list of installed fonts with given name:
  
  `fc-list | grep 'DejaVu Serif'`

**fc-match**

Match available fonts.

- Return a sorted list of best matching fonts:
  
  `fc-match -s 'DejaVu Serif'`

**fc-pattern**

Shows information about a font matching a pattern.

- Display default information about a font:
  
  `fc-pattern -d 'DejaVu Serif'`

**figlet**

Generate ASCII banners from user input.

- Generate by directly inputting text:
  
  `figlet input_text`

- Use a custom font file:
  
  `figlet input_text -f font_file_name`

- Pipe command output through figlet:
  
  `command | figlet`
findmnt

Find your filesystem.

- List all mounted filesystems:
  findmnt

- Search for a device:
  findmnt /dev/sdb1

- Search for a mountpoint:
  findmnt /

- Find filesystems in specific type:
  findmnt -t ext4

- Find filesystems with specific label:
  findmnt LABEL=BigStorage

firewall-cmd

The firewalld command line client.

- View the available firewall zones:
  firewall-cmd --get-active-zones

- View the rules which are currently applied:
  firewall-cmd --list-all

- Permanently open the port for a service in the specified zone (like port 443 when in the public zone):
  firewall-cmd --permanent --zone=public --add-service=https

- Permanently close the port for a service in the specified zone (like port 80 when in the public zone):
  firewall-cmd --permanent --zone=public --remove-service=http

- Reload firewalld to force rule changes to take effect:
  firewall-cmd --reload
**free**

Display amount of free and used memory in the system.

- Display system memory:
  
  ```sh
tree
  ```

- Display memory in Bytes/KB/MB/GB:
  
  ```sh
tree -b/-k/-m/-g
  ```

- Display memory in human readable units:
  
  ```sh
tree -h
  ```

- Continuous monitor memory (refresh every X seconds):
  
  ```sh
tree -s X
  ```

**fuser**

Display process IDs currently using files or sockets. Require admin privileges.

- Identify process using a TCP socket:
  
  ```sh
tree -n tcp port
  ```

**getent**

Get entries from Name Service Switch libraries.

- Get list of all groups:
  
  ```sh
tree group
  ```

- See the members of a group:
  
  ```sh
tree group group_name
  ```

- Get list of all services:
  
  ```sh
tree services
  ```

- Find a username by UID:
  
  ```sh
tree passwd 1000
  ```

- Perform a reverse DNS lookup:
  
  ```sh
tree hosts host
  ```
**groupadd**

Add user groups to the system.

- Create a new Linux group:
  
  `groupadd group_name`

- Create new group with a specific groupid:
  
  `groupadd group_name -g group_id`

**groupdel**

Delete existing user groups from the system.

- Delete an existing group:
  
  `groupdel group_name`

**groupmod**

Modify existing user groups in the system.

- Change the group name:
  
  `groupmod -n new_group_name old_group_name`

- Change the group id:
  
  `groupmod -g new_group_id old_group_name`

**halt**

Power off or reboot the machine.

- Power the machine off:
  
  `halt`

- Reboot the machine:
  
  `halt --reboot`
**head**

Output the first part of files.

- Output the first few lines of a file:
  
  head -n count_of_lines filename

- Output the first few bytes of a file:
  
  head -c size_in_bytes filename

- Output everything but the last few lines of a file:
  
  head -n -count_of_lines filename

- Output everything but the last few bytes of a file:
  
  head -c -size_in_bytes filename

**hostname**

Show or set the system’s host name.

- Show current host name:
  
  hostname

- Show the network address of the host name:
  
  hostname -i

- Show all network addresses of the host:
  
  hostname -I

- Show the FQDN (Fully Qualified Domain Name):
  
  hostname --fqdn

- Set current host name:
  
  hostname new_hostname
**htop**

Display dynamic real-time information about running processes. An enhanced version of top.

- Start htop:
  ```
  htop
  ```

- Start htop displaying only processes owned by given user:
  ```
  htop -u user_name
  ```

- Get help about interactive commands:
  ```
  ?
  ```

**hwclock**

Used for reading or changing the hardware clock. Usually requires root.

- Display the current time as reported by the hardware clock:
  ```
  hwclock
  ```

- Write the current software clock time to the hardware clock (sometimes used during system setup):
  ```
  hwclock --systohc
  ```

- Write the current hardware clock time to the software clock:
  ```
  hwclock --hctosys
  ```

**ifdown**

Disable network interfaces.

- Disable interface eth0:
  ```
  ifdown eth0
  ```

- Disable all interfaces which are enabled:
  ```
  ifdown -a
  ```
**ifup**

Tool used to enable network interfaces.

- Enable interface eth0:
  
  ```
  ifup eth0
  ```

- Enable all the interfaces defined with “auto” in /etc/network/interfaces:
  
  ```
  ifup -a
  ```

**iostat**

Report statistics for devices and partitions.

- Display a report of CPU and disk statistics since system startup:
  
  ```
  iostat
  ```

- Display a report of CPU and disk statistics with units converted to megabytes:
  
  ```
  iostat -m
  ```

- Display CPU statistics:
  
  ```
  iostat -c
  ```

- Display disk statistics with disk names (including LVM):
  
  ```
  iostat -N
  ```

- Display extended disk statistics with disk names for device “sda”:
  
  ```
  iostat -xN sda
  ```

- Display incremental reports of CPU and disk statistics every 2 seconds:
  
  ```
  iostat 2
  ```
**ip**

Show / manipulate routing, devices, policy routing and tunnels.

- List interfaces with detailed info:
  
  ip a

- Display the routing table:
  
  ip r

- Make an interface up/down:
  
  ip link set interface up/down

- Add/Delete an ip address to an interface:
  
  ip addr add/del ip/mask dev interface

- Add an default route:
  
  ip route add default via ip dev interface

**iptables**

Program that allows to configure tables, chains and rules provided by the Linux kernel firewall.

- See chains and rules for specific table:
  
  sudo iptables -t table_name -vnL

- Set chain policy rule:
  
  sudo iptables -p chain rule

- Append rule to chain policy for IP:
  
  sudo iptables -A chain -s ip -j rule

- Append rule to chain policy for IP considering protocol and port:
  
  sudo iptables -A chain -s ip -p protocol --dport port -j rule

- Delete chain rule:
  
  sudo iptables -D chain rule_line_number

- Save iptables configuration:
  
  sudo iptables-save > path/to/iptables_file
**jobs**

BASH builtin for viewing information about processes spawned by the current shell.

- View jobs spawned by the current shell:
  ```
  jobs
  ```
- List jobs and their process ids:
  ```
  jobs -l
  ```
- Display information about jobs with changed status:
  ```
  jobs -n
  ```
- Display process id of process group leader:
  ```
  jobs -p
  ```
- Display running processes:
  ```
  jobs -r
  ```
- Display stopped processes:
  ```
  jobs -s
  ```

**journalctl**

Query the systemd journal.

- Show all messages from this boot:
  ```
  journalctl -b
  ```
- Show all messages from last boot:
  ```
  journalctl -b -1
  ```
- Follow new messages (like `tail -f` for traditional syslog):
  ```
  journalctl -f
  ```
- Show all messages by a specific unit:
  ```
  journalctl -u unit
  ```
- Show all messages by a specific process:
  ```
  journalctl _PID=pid
  ```
- Show all messages by a specific executable:
  ```
  journalctl /path/to/executable
  ```
locate

Find filenames quickly.

- Look for pattern in the database. Note: the database is recomputed periodically (usually weekly or daily):

  `locate pattern`

- Look for a file by its exact filename (a pattern containing no globbing characters is interpreted as *pattern*):

  `locate */filename`

- Recompute the database. You need to do it if you want to find recently added files:

  `sudo updatedb`

logger

Add messages to syslog (/var/log/syslog).

- Log a message to syslog:

  `logger message`

- Take input from stdin and log to syslog:

  `echo log_entry | logger`

- Send the output to a remote syslog server running at a given port. Default port is 514:

  `echo log_entry | logger --server hostname --port port`

- Use a specific tag for every line logged. Default is the name of logged in user:

  `echo log_entry | logger --tag tag`

- Log messages with a given priority. Default is user.notice. See man `logger` for all priority options:

  `echo log_entry | logger --priority user.warning`
**lsattr**

List file attributes on a Linux file system.

- Display the attributes of the files in the current directory:
  
  `lsattr`

- List the attributes of files in a particular path:
  
  `lsattr path`

- List file attributes recursively in the current and subsequent directories:
  
  `lsattr -R`

- Show attributes of all the files in the current directory, including hidden ones:
  
  `lsattr -a`

- Display attributes of directories in the current directory:
  
  `lsattr -d`

**lsb_release**

Provides certain LSB (Linux Standard Base) and distribution-specific information.

- Print all available information:
  
  `lsb_release -a`

- Print a description (usually the full name) of the operating system:
  
  `lsb_release -d`

- Print only the operating system name (ID), suppressing the field name:
  
  `lsb_release -i -s`

- Print the release number and codename of the distribution, suppressing the field names:
  
  `lsb_release -rcs`
**lsblk**

Lists information about devices.

- List all storage devices in a tree-like format:
  
  lsblk

- Also list empty devices:
  
  lsblk -a

- Print the SIZE column in bytes rather than in a human-readable format:
  
  lsblk -b

- Output info about filesystems:
  
  lsblk -f

- Use ASCII characters for tree formatting:
  
  lsblk -i

- Output info about block-device topology:
  
  lsblk -t

**ltrace**

Display dynamic library calls of a process.

- Print (trace) library calls of a program binary:
  
  ltrace ./program

- Count library calls. Print a handy summary at the bottom:
  
  ltrace -c /path/to/program

- Trace calls to malloc and free, omit those done by libc:
  
  ltrace -e malloc+free-@libc.so* /path/to/program

- Write to file instead of terminal:
  
  ltrace -o file /path/to/program
**md5sum**

Calculate MD5 cryptographic checksums.

- Calculate the MD5 checksum for a file:
  
  `md5sum filename1`

- Calculate MD5 checksums for multiple files:
  
  `md5sum filename1 filename2`

- Read a file of MD5SUMs and verify all files have matching checksums:
  
  `md5sum -c filename.md5`

**mdadm**

RAID management utility.

- Create array:
  
  `mdadm --create /path/to/raid_device_file --level raid_level --raid-devices number_of_disks /path/to/disk_device_file`

- Stop array:
  
  `mdadm -S /path/to/raid_device_file`

- Mark disk as failed:
  
  `mdadm /path/to/raid_device_file -f /path/to/disk_device_file`

- Remove disk:
  
  `mdadm /path/to/raid_device_file -r /path/to/disk_device_file`

- Add disk to array:
  
  `mdadm /path/to/raid_device_file -a /path/to/disk_device_file`

- Show RAID info:
  
  `mdadm -D /path/to/raid_device_file`
mke2fs
Creates a Linux filesystem inside a partition.
- Create an ext2 filesystem in partition 1 of device b (sdb1):
  mkfs.ext2 /dev/sdb1
- Create an ext3 filesystem in partition 1 of device b (sdb1):
  mkfs.ext3 /dev/sdb1
- Create an ext3 filesystem in partition 1 of device b (sdb1):
  mkfs.ext3 /dev/sdb1

mkfs.cramfs
Creates a ROM filesystem inside a partition.
- Create a ROM filesystem inside partition 1 on device b (sdb1):
  mkfs.cramfs /dev/sdb1
- Create a ROM filesystem with a volume-name:
  mkfs.cramfs -n volume_name /dev/sdb1

mkfs.exfat
Creates an exfat filesystem inside a partition.
- Create an exfat filesystem inside partition 1 on device b (sdb1):
  mkfs.exfat /dev/sdb1
- Create filesystem with a volume-name:
  mkfs.exfat -n volume_name /dev/sdb1
- Create filesystem with a volume-id:
  mkfs.exfat -i volume_id /dev/sdb1
**mkfs.fat**

Creates an MS-DOS filesystem inside a partition.

- Create a fat filesystem inside partition 1 on device b (sdb1):
  ```
  mkfs.fat /dev/sdb1
  ```

- Create filesystem with a volume-name:
  ```
  mkfs.fat -n volume_name /dev/sdb1
  ```

- Create filesystem with a volume-id:
  ```
  mkfs.fat -i volume_id /dev/sdb1
  ```

- Use 5 instead of 2 file allocation tables:
  ```
  mkfs.fat -f 5 /dev/sdb1
  ```

**mkfs.minix**

Creates a Minix filesystem inside a partition.

- Create a Minix filesystem inside partition 1 on device b (sdb1):
  ```
  mkfs.minix /dev/sdb1
  ```

**mkfs.ntfs**

Creates a NTFS filesystem inside a partition.

- Create a NTFS filesystem inside partition 1 on device b (sdb1):
  ```
  mkfs.ntfs /dev/sdb1
  ```

- Create filesystem with a volume-label:
  ```
  mkfs.ntfs -L volume_label /dev/sdb1
  ```

- Create filesystem with specific UUID:
  ```
  mkfs.ntfs -U UUID /dev/sdb1
  ```

221
**mkfs.vfat**

Creates an MS-DOS filesystem inside a partition.

- Create a.vfat filesystem inside partition 1 on device b (sdb1):
  
  `mkfs.vfat /dev/sdb1`

- Create filesystem with a volume-name:
  
  `mkfs.vfat -n volume_name /dev/sdb1`

- Create filesystem with a volume-id:
  
  `mkfs.vfat -i volume_id /dev/sdb1`

- Use 5 instead of 2 file allocation tables:
  
  `mkfs.vfat -f 5 /dev/sdb1`

**nethogs**

Monitor bandwidth usage per process.

- Start nethogs as root (default device is eth0):
  
  `sudo nethogs`

- Monitor bandwidth on specific device:
  
  `sudo nethogs device`

- Monitor bandwidth on multiple devices:
  
  `sudo nethogs device1 device2`

- Specify refresh rate:
  
  `sudo nethogs -t seconds`
**netstat**

Displays various networks related information such as open connections, open socket ports etc.

- List all ports:
  ```
  netstat -a
  ```
- List all listening ports:
  ```
  netstat -l
  ```
- List listening TCP ports:
  ```
  netstat -t
  ```
- Display PID and program names:
  ```
  netstat -p
  ```
- List information continuously:
  ```
  netstat -c
  ```
- List routes and do not resolve IP to hostname:
  ```
  netstat -rn
  ```
- List listening TCP and UDP ports (+ user and process if you’re root):
  ```
  netstat -lepunt
  ```

**nm**

List symbol names in object files.

- List global (extern) functions in a file (prefixed with T):
  ```
  nm -g file.o
  ```
- Demangle C++ symbols (make them readable):
  ```
  nm --demangle file.o
  ```
- List only undefined symbols in a file:
  ```
  nm -u file.o
  ```
- List all symbols, even debugging symbols:
  ```
  nm -a file.o
  ```
**notify-send**

Uses the current desktop environment’s notification system to create a notification.

- Show a notification with the title “Test” and the content “This is a test”:
  ```
  notify-send "Test" "This is a test"
  ```
- Show a notification with a custom icon:
  ```
  notify-send -i icon.png "Test" "This is a test"
  ```
- Show a notification for 5 seconds:
  ```
  notify-send -t 5000 "Test" "This is a test"
  ```

**pacman**

Arch Linux package manager utility.

- Synchronize and update all packages:
  ```
  pacman -Syu
  ```
- Install a new package:
  ```
  pacman -S package_name
  ```
- Remove a package and its dependencies:
  ```
  pacman -Rs package_name
  ```
- Search the package database for a regular expression or keyword:
  ```
  pacman -Ss "search_pattern"
  ```
- List installed packages and versions:
  ```
  pacman -Q
  ```
- List only the explicitly installed packages and versions:
  ```
  pacman -Qe
  ```
- Find which package owns a certain file:
  ```
  pacman -Qo filename
  ```
- Empty package cache to free up space:
  ```
  pacman -Scc
  ```
pkgadd

Add a package to a CRUX system.

- Install a local software package:
  pkgadd package_name

- Update an already installed package from a local package:
  pkgadd -u package_name

pkginfo

Query the package database on a CRUX system.

- List installed packages and their versions:
  pkginfo -i

- List files owned by a package:
  pkginfo -l package_name

- List the owner(s) of files matching a pattern:
  pkginfo -o pattern

- Print the footprint of a file:
  pkginfo -f file

pkgmk

Make a binary package for use with pkgadd on CRUX.

- Make and download a package:
  pkgmk -d

- Install the package after making it:
  pkgmk -d -i

- Upgrade the package after making it:
pkgmk -d -u
- Ignore the footprint when making a package:
  pkgmk -d -if
- Ignore the MD5 sum when making a package:
  pkgmk -d -im
- Update the package’s footprint:
  pkgmk -uf

pkgrm

Remove a package from a CRUX system.
- Remove an installed package:
  pkgrm package_name

ports

Update/list the ports tree on a CRUX system.
- Update the ports tree:
  ports -u
- List the ports in the current tree:
  ports -l
- Check the differences between installed packages and the ports tree:
  ports -d

print

An alias to a run-mailcap’s action print. Originally run-mailcap is used to process mime-type/file.
- Print action can be used to print any file on default run-mailcap tool:
  print filename
- With run-mailcap:
  run-mailcap --action=print filename
**prt-get**

The advanced CRUX package manager.

- Install a package:
  
  ```
  prt-get install package_name
  ```

- Install a package with dependency handling:
  
  ```
  prt-get depinst package_name
  ```

- Update a package manually:
  
  ```
  prt-get upgrade package_name
  ```

- Remove a package:
  
  ```
  prt-get remove package_name
  ```

- Upgrade the system from the local ports tree:
  
  ```
  prt-get sysup
  ```

- Search the ports tree:
  
  ```
  prt-get search package_name
  ```

- Search for a file in a package:
  
  ```
  prt-get fsearch file
  ```

**pvcreate**

Initialize a disk or partition for use by LVM (Logical Volume Manager).

- Initialize `/dev/sda1` to use lvm:
  
  ```
  pvcreate /dev/sda1
  ```

- Force the creation without any confirmation:
  
  ```
  pvcreate --force /dev/sda1
  ```
**pwgen**

Generate pronounceable passwords.

- Generate random password with symbols:
  
  ```bash
  pwgen -y length
  ```

- Generate secure, hard-to-memorize passwords:
  
  ```bash
  pwgen -s length
  ```

- Generate password with at least one capital letter in them:
  
  ```bash
  pwgen -c length
  ```

**quotacheck**

Scan a filesystem for disk usage; create, check and repair quota files. It is best to run quota check with quotas turned off to prevent damage or loss to quota files.

- Check quotas on all mounted non-NFS filesystems:
  
  ```bash
  sudo quotacheck --all
  ```

- Force check even if quotas are enabled (this can cause damage or loss to quota files):
  
  ```bash
  sudo quotacheck --force mountpoint
  ```

- Check quotas on a given filesystem in debug mode:
  
  ```bash
  sudo quotacheck --debug mountpoint
  ```

- Check quotas on a given filesystem, displaying the progress:
  
  ```bash
  sudo quotacheck --verbose mountpoint
  ```

- Check user quotas:
  
  ```bash
  sudo quotacheck --user user mountpoint
  ```

- Check group quotas:
  
  ```bash
  sudo quotacheck --group group mountpoint
  ```
**rdesktop**

Remote Desktop Protocol client. It can be used to connect the remote computer using the RDP protocol.

- Connect to a remote computer (default port is 3389):
  
rdesktop -u username -p password host:port

- Simple Examples:
  
rdesktop -u Administrator -p passwd123 192.168.1.111:3389

- Connect to a remote computer with full screen (press Ctrl+Alt+Enter to exist):
  
rdesktop -u username -p password -f host:port

- Use the customed resolution (use the letter 'x' between the number):
  
rdesktop -u username -p password -g 1366x768 host:port

- Connect to a remote computer using domain user:
  
rdesktop -u username -p password -d domainname host:port

- Use the 16 bit color (speed up):
  
rdesktop -u username -p password -a 16 host:port

**reboot**

Reboot the system.

- Reboot immediately:
  
  reboot

- Reboot immediately without gracefully shutdown:
  
  reboot -f
**repquota**

Display a summary of existing file quotas for a filesystem.

- Report stats for all quotas in use:
  
  ```
  sudo repquota -all
  ```

- Report quota stats for all users, even those who aren’t using any of their quota:
  
  ```
  sudo repquota -v filesystem
  ```

- Report on quotas for users only:
  
  ```
  repquota --user filesystem
  ```

- Report on quotas for groups only:
  
  ```
  sudo repquota --group filesystem
  ```

- Report on used quota and limits in a human-readable format:
  
  ```
  sudo repquota --human-readable filesystem
  ```

- Report on all quotas for users and groups in a human-readable format:
  
  ```
  sudo repquota -augs
  ```

**rpm**

RPM Package Manager.

- Show version of httpd package:
  
  ```
  rpm -q httpd
  ```

- List versions of all matching packages:
  
  ```
  rpm -qa 'mariadb*' 
  ```

- Identify owner of a file and show version of the package:
  
  ```
  rpm -qf /etc/postfix/main.cf
  ```

- List package-owned files:
  
  ```
  rpm -ql kernel
  ```

- Show scriptlets from an RPM file:
  
  ```
  rpm -qP --scripts some.rpm
  ```

- Show changed, missing and/or incorrectly installed files of matching packages:
  
  ```
  rpm -V a 'php-*'
  ```
**run-mailcap**

Run MailCap Programs. Run mailcap view, see, edit, compose, print - execute programs via entries in the mailcap file (or any of its aliases) will use the given action to process each mime-type/file.

- Individual actions/programs on run-mailcap can be invoked with action flag:
  
  ```
  run-mailcap --action=ACTION [--option[=value]]
  ```
  
- In simple language:
  
  ```
  run-mailcap --action=ACTION filename
  ```
  
- Turn on extra information:
  
  ```
  run-mailcap --action=ACTION --debug filename
  ```
  
- Ignore any “copiousoutput” directive and forward output to STD□OUT:
  
  ```
  run-mailcap --action=ACTION --nopager filename
  ```
  
- Display the found command without actually executing it:
  
  ```
  run-mailcap --action=ACTION --norun filename
  ```

**see**

Alias to run-mailcap’s view. An alias to a run-mailcap’s action print.

- See action can be used to view any file (usually image) on default mailcap explorer:
  
  ```
  see filename
  ```
  
- Using with run-mailcap:
  
  ```
  run-mailcap --action=view filename
  ```
service

Manage services by running init scripts. The full script path should be omitted (/etc/init.d/ is assumed).

- Start/Stop/Restart/Reload service (start/stop should always be available):
  
  ```
  service init_script start|stop|restart|reload
  ```

- Do a full restart (runs script twice with start and stop):

  ```
  service init_script --full-restart
  ```

- Show the current status of a service:

  ```
  service init_script status
  ```

- List the status of all services:

  ```
  service --status-all
  ```

setfacl

Set file access control lists (ACL).

- Modify ACL of a file for user with read and write access:

  ```
  setfacl -m u:username:rw file
  ```

- Modify default ACL of a file for all users:

  ```
  setfacl -d -m u::rw file
  ```

- Remove ACL of a file for an user:

  ```
  setfacl -x u:username file
  ```

- Remove all ACL entries of a file:

  ```
  setfacl -b file
  ```
**sha1sum**

Calculate SHA1 cryptographic checksums.

- Calculate the SHA1 checksum for a file:
  
  `sha1sum filename1`

- Calculate SHA1 checksums for multiple files:

  `sha1sum filename1 filename2`

- Read a file of SHA1 sums and verify all files have matching checksums:

  `sha1sum -c filename.sha1`

**sha224sum**

Calculate SHA224 cryptographic checksums.

- Calculate the SHA224 checksum for a file:

  `sha224sum filename1`

- Calculate SHA224 checksums for multiple files:

  `sha224sum filename1 filename2`

- Read a file of SHA224 sums and verify all files have matching checksums:

  `sha224sum -c filename.sha224`

**sha256sum**

Calculate SHA256 cryptographic checksums.

- Calculate the SHA256 checksum for a file:

  `sha256sum filename1`

- Calculate SHA256 checksums for multiple files:

  `sha256sum filename1 filename2`

- Read a file of SHA256 sums and verify all files have matching checksums:

  `sha256sum -c filename.sha256`
sha384sum

Calculate SHA384 cryptographic checksums.
- Calculate the SHA384 checksum for a file:
  sha384sum filename1
- Calculate SHA384 checksums for multiple files:
  sha384sum filename1 filename2
- Read a file of SHA384 sums and verify all files have matching checksums:
  sha384sum -c filename.sha384

sha512sum

Calculate SHA512 cryptographic checksums.
- Calculate the SHA384 checksum for a file:
  sha512sum filename1
- Calculate SHA384 checksums for multiple files:
  sha512sum filename1 filename2
- Read a file of SHA512 sums and verify all files have matching checksums:
  sha512sum -c filename.sha512

shuf

Generate random permutations.
- Randomize the order of lines in a file and output the result:
  shuf filename
- Only output the first n entries of the result:
  shuf -n n filename
- Write output to another file:
  shuf -o another_filename filename
- Generate random numbers in range:
  shuf -i low-high
**shutdown**

Shutdown and reboot the system.

- Power off (halt) immediately:
  
  `shutdown -h now`

- Reboot immediately:
  
  `shutdown -r now`

- Reboot in 5 minutes:
  
  `shutdown -r +5 &`

- Shutdown at 1:00 pm (Uses 24h clock):
  
  `shutdown -h 13:00`

- Cancel a pending shutdown/reboot operation:
  
  `shutdown -c`

**sort**

Sort lines of text files.

- Sort a file in ascending order:
  
  `sort filename`

- Sort a file in descending order:
  
  `sort -r filename`

- Sort a file using numeric rather than alphabetic order:
  
  `sort -n filename`

- Sort the passwd file by the 3rd field, numerically:
  
  `sort -t: -k 3n /etc/passwd`

- Sort human-readable numbers (in this case the 5th field of `ls -lh`):
  
  `ls -lh | sort -h -k 5`
**ss**

Utility to investigate sockets.

- Show all TCP/UDP/RAW/UNIX sockets:
  
  ```
  ss -a -t|u|w|x
  ```

- Filter TCP sockets by states, only/exclude:
  
  ```
  ss state/exclude bucket/big/connected/synchronized/...
  ```

- Show all TCP sockets connected to the local HTTPS port (443):
  
  ```
  ss -t src :443
  ```

- Show all TCP sockets along with processes connected to a remote ssh port:
  
  ```
  ss -pt dst :ssh
  ```

- Show all UDP sockets connected on specific source and destination ports:
  
  ```
  ss -u 'sport == :source_port and dport == :destination_port'
  ```

- Show all TCP IPv4 sockets locally connected on the subnet 192.168.0.0/16:
  
  ```
  ss -4t src 192.168/16
  ```

**strace**

Troubleshooting tool for tracing system calls.

- Start tracing a specific process by its PID:
  
  ```
  strace -p pid
  ```

- Trace a process and filter output by system call:
  
  ```
  strace -p pid -e system call name
  ```

- Count time, calls, and errors for each system call and report a summary on program exit:
  
  ```
  strace -p pid -c
  ```

- Show the time spent in every system call:
  
  ```
  strace -p pid -T
  ```

- Start tracing a program by executing it:
  
  ```
  strace program
  ```
**sysctl**

List and change kernel runtime variables.

- Show all available variables and their values:
  
  ```
  sysctl -a
  ```

- Set a changeable kernel state variable:

  ```
  sysctl -w section.tunable=value
  ```

- Get currently open file handlers:

  ```
  sysctl fs.file-nr
  ```

- Get limit for simultaneous open files:

  ```
  sysctl fs.file-max
  ```

- Apply changes from /etc/sysctl.conf:

  ```
  sysctl -p
  ```

**systemctl**

Control the systemd system and service manager.

- List failed units:

  ```
  systemctl --failed
  ```

- Start/Stop/Restart/Reload a service:

  ```
  systemctl start/stop/restart/reload unit
  ```

- Show the status of a unit:

  ```
  systemctl status unit
  ```

- Enable/Disable a unit to be started on bootup:

  ```
  systemctl enable/disable unit
  ```

- Mask/Unmask a unit, prevent it to be started on bootup:

  ```
  systemctl mask/unmask unit
  ```

- Reload systemd, scanning for new or changed units:

  ```
  systemctl daemon-reload
  ```
**systemd-analyze**

Show timing details about the boot process of units (services, mount points, devices, sockets).

- List time of each unit to start up:
  
  ```
  systemctl-analyze blame
  ```

- Print a tree of the time critical chain of units:
  
  ```
  systemctl-analyze critical-chain
  ```

**tcpflow**

Capture TCP traffic for debugging and analysis.

- Show all data on the given interface and port:
  
  ```
  tcpflow -c -i eth0 port 80
  ```

**timedatectl**

Control the system time and date.

- To check the current system clock time:
  
  ```
  timedatectl
  ```

- To set the local time of the system clock directly:
  
  ```
  timedatectl set-time "yyyy-MM-dd hh:mm:ss"
  ```

- To list available timezones:
  
  ```
  timedatectl list-timezones
  ```

- To change timezones:
  
  ```
  timedatectl set-timezone timezone
  ```

- To enable Network Time Protocol (NTP) syncing:
  
  ```
  timedatectl set-ntp on
  ```
**top**

Display dynamic real-time information about running processes.

- Start top:
  ```
  top
  ```

- Do not show any idle or zombie processes:
  ```
  top -i
  ```

- Show only processes owned by given user:
  ```
  top -u user_name
  ```

- Show only the processes with the given PID(s), passed as a comma-separated list. (Normally you wouldn’t know PIDs off hand. This example picks the PIDs from the process name):
  ```
  top -p $(pgrep -d ',' process_name)
  ```

- Get help about interactive commands:
  ```
  ?
  ```

**tree**

Show the contents of the current directory as a tree.

- Show files and directories up to ‘num’ levels of depth (where 1 means the current directory):
  ```
  tree -L num
  ```

- Show directories only:
  ```
  tree -d
  ```

- Show hidden files too:
  ```
  tree -a
  ```

- Print the tree without indentation lines, showing the full path instead (use -N to not escape whitespace and special characters):
  ```
  tree -i -f
  ```

- Print the size of each node next to it, in human-readable format:
  ```
  tree -s -h
  ```

- Filter the tree using a wildcard (glob) pattern:
  ```
  tree -P *.txt
  ```
ufw

Uncomplicated Firewall. Frontend for iptables aiming to make configuration of a firewall easier.

- Enable ufw:
  ufw enable
- Disable ufw:
  ufw disable
- Show ufw rules, along with their numbers:
  ufw status numbered
- Allow incoming traffic on port 5432 on this host:
  ufw allow 5432
- Allow only TCP traffic from 192.168.0.4 to any address on this host, on port 22:
  ufw allow proto tcp from 192.168.0.4 to any port 22
- Deny traffic on port 80 on this host:
  ufw deny 80
- Deny all UDP traffic to port 22:
  ufw deny proto udp from any to any port 22
- Delete a particular rule. The rule number can be retrieved from the ufw status numbered command:
  ufw delete rule_number

ulimit

Get and set user limits.

- Get the properties of all the user limits:
  ulimit -a
- Get hard limit for the number of simultaneously opened files:
  ulimit -H -n
- Get soft limit for the number of simultaneously opened files:
  ulimit -S -n
- Set max per-user process limit:
  ulimit -u 30
**umask**

Manage the read/write/execute permissions that are masked out (i.e. restricted) for newly created files by the user.

- Display the current mask in octal notation:

  \texttt{umask}
  
- Display the current mask in symbolic (human-readable) mode:

  \texttt{umask -S}
  
- Change the mask symbolically to allow read permission for all users (the rest of the mask bits are unchanged):

  \texttt{umask a+r}
  
- Set the mask (using octal) to restrict no permissions for the file’s owner, and restrict all permissions for everyone else:

  \texttt{umask 077}

**uname**

Print details about the current machine and the operating system running on it. Note: for additional information about the operating system, try the \texttt{lsb_release} command.

- Print hardware-related information: machine and processor:

  \texttt{uname -mp}
  
- Print software-related information: operating system, release number, and version:

  \texttt{uname -srv}
  
- Print the nodename (hostname) of the system:

  \texttt{uname -n}
  
- Print all available system information (hardware, software, nodename):

  \texttt{uname -a}
**unexpand**

Convert spaces to tabs.
- Convert blanks in each file to tabs, writing to standard output:
  ```bash
  unexpand file
  ```
- Convert blanks to tabs, reading from standard output:
  ```bash
  unexpand
  ```
- Convert all blanks, instead of just initial blanks:
  ```bash
  unexpand -a file
  ```
- Convert only leading sequences of blanks (overrides -a):
  ```bash
  unexpand --first-only file
  ```
- Have tabs a certain number of characters apart, not 8 (enables -a):
  ```bash
  unexpand -t number file
  ```

**useradd**

Create a new user.
- Create new user:
  ```bash
  useradd name
  ```
- Create new user with a default home directory:
  ```bash
  useradd --create-home name
  ```
- Create new user with specified shell:
  ```bash
  useradd --shell /path/to/shell name
  ```
- Create new user belonging to additional groups (mind the lack of whitespace):
  ```bash
  useradd --groups group1,group2 name
  ```
- Create new system user without a home directory:
  ```bash
  useradd --no-create-home --system name
  ```
userdel

Remove a user.
- Remove a user and their home directory:
  userdel -r name

usermod

Modifies a user account.
- Change a user's name:
  usermod -l newname user
- Add user to supplementary groups (mind the whitespace):
  usermod -a -G group1,group2 user
- Create a new home directory for a user and move their files to it:
  usermod -m -d /path/to/home user

vgcreate

Create a volume group.
- Create a new volume group called vg1 using /dev/sda1:
  vgcreate vg1 /dev/sda1
- Create a new volume group called vg1 using multiple devices:
  vgcreate vg1 /dev/sda1 /dev/sdb1 /dev/sdc1

wall

Write a message on the terminals of users currently logged in. Only available to super-user.
- Send a message:
  echo "message" | wall
- Send a message from a file:
  wall file
- Send a message with timeout (default 300):
  wall -t seconds file
**watch**

Execute a command repeatedly, and monitor the output in full-screen mode.

- Monitor files in the current folder:
  ```
  watch ls
  ```
- Monitor disk space and highlight the changes:
  ```
  watch -d df
  ```
- Monitor "node" processes, refreshing every 3 seconds:
  ```
  watch -n 3 "ps aux | grep node"
  ```

**whatis**

Display one-line descriptions from manual pages.

- Display a description from a man page:
  ```
  whatis command
  ```
- Don’t cut the description off at the end of the line:
  ```
  whatis --long command
  ```
- Search man page descriptions with a regular expression:
  ```
  whatis --regex regular_expression
  ```

**wpa_cli**

Add and configure wlan interfaces.

- Scan for available networks:
  ```
  wpa_cli scan
  ```
- Show scan results:
  ```
  wpa_cli scan_results
  ```
- Add a network:
wpa_cli add_network number

- Set a network’s SSID:
  wpa_cli set_network number ssid "SSID"

- Enable network:
  wpa_cli enable_network number

- Save config:
  wpa_cli save_config

x11vnc

A VNC server that will enable VNC on an existing display. By default, once a client disconnects the server will terminate.

- Launch a VNC server that allows multiple clients to connect:
  x11vnc -shared

- Launch the server where the user can only view the screen, and will continue to run even after the last client disconnects:
  x11vnc -forever -viewonly

- Launch a VNC server on a specific display and screen:
  x11vnc -display :screen.display

- Launch a VNC server on screen 2 with the default display:
  x11vnc -display :2

- Launch a VNC server on the second monitor:
  x11vnc -display :0.1
**xclip**

Copy STDIN to clipboard or print clipboard to STDOUT.

- Copy output to clipboard:
  ```
  echo 123 | xclip -i
  ```

- Copy output to system clipboard:
  ```
  echo 123 | xclip -sel clip
  ```

- Paste clipboard:
  ```
  xclip -o > file.txt
  ```

**xdotool**

Command line automation for X11.

- Retrieve the X-Windows window ID of the running Firefox window(s):
  ```
  xdotool search --onlyvisible --name firefox
  ```

- Click the right mouse button:
  ```
  xdotool click 3
  ```

**xeyes**

Display eyes on the screen that follow the mouse cursor.

- Launch xeyes on local display:
  ```
  xeyes
  ```

- Launch xeyes on a remote display 0 screen 0:
  ```
  xeyes -display remote_host:0.0
  ```
**xinput**

List available input devices, query information about a device and change input device settings.

- List all input devices:
  
  `xinput list`

- Disconnect an input from its master:

  `xinput float id`

- Reattach an input as slave to a master:

  `xinput reattach id master_id`

**xrandr**

Set the size, orientation and/or reflection of the outputs for a screen.

- Display the current state of the system (known screens, resolutions, ...):

  `xrandr --query`

- Disable disconnected outputs and enable connected ones with default settings:

  `xrandr --auto`

- Change the resolution and update frequency of DisplayPort 1 to 1920x1080, 60Hz:

  `xrandr --output DP1 --mode 1920x1080 --rate 60`

- Set the resolution of HDMI2 to 1280x1024 and put it on the right of DP1:

  `xrandr --output HDMI2 --mode 1280x1024 --right-of DP1`

- Disable the VGA1 output:

  `xrandr --output VGA1 --off`
**xsel**

X11 selection and clipboard manipulation tool.

- Use a command’s output as input of the clipboard (equivalent to Ctrl+C):
  
  ```
  echo 123 | xsel -ib
  ```

- Use the contents of a file as input of the clipboard:
  
  ```
  cat file | xsel -ib
  ```

- Output the clipboard’s contents into the terminal (equivalent to Ctrl+V):
  
  ```
  xsel -ob
  ```

- Output the clipboard’s contents into a file:
  
  ```
  xsel -ob > file
  ```

- Clear the clipboard:
  
  ```
  xsel -cb
  ```

- Output the X11 primary selection’s contents into the terminal (equivalent to a mouse middle-click):
  
  ```
  xsel -op
  ```

**xsetwacom**

Command line tool to change settings for Wacom pen tablets at runtime.

- List all the available wacom devices. The device name is in the first column:
  
  ```
  xsetwacom list
  ```

- Set Wacom area to specific screen. Get name of the screen with `xrandr`:
  
  ```
  xsetwacom set "device name" MapToOutput screen
  ```

- Set mode to relative (like a mouse) or absolute (like a pen) mode:
  
  ```
  xsetwacom set "device name" Mode "Relative|Absolute"
  ```

- Rotate the input (useful for tablet-PC when rotating screen) by 0|90|180|270 degrees from "natural" rotation:
  
  ```
  xsetwacom set "device name" Rotate none|half|cw|ccw
  ```

- Set button to only work when the tip of the pen is touching the tablet:
  
  ```
  xsetwacom set "device name" TabletPCButton "on"
  ```
yaourt

Arch Linux utility for building packages from the Arch User Repository.

- Synchronize and update all packages (including AUR):
  yaourt -Syua
- Install a new package (includes AUR):
  yaourt -S package-name
- Remove a package and its dependencies (includes AUR packages):
  yaourt -Rs package-name
- Search the package database for a keyword (including AUR):
  yaourt -Ss package-name
- List installed packages, versions, and repositories (AUR packages will be listed under the repository name 'local'):
  yaourt -Q

yum

Package management utility for RHEL, Fedora, and CentOS (for older versions).

- Synchronize list of packages and versions available. This should be run first, before running subsequent yum commands:
  yum update
- Install a new package:
  yum install package
- Install a new package and assume yes to all questions (also works with update, great for automated updates):
  yum -y install package
- Find the package that provides a particular command:
  yum provides command
- Remove a package:
  yum remove package
- Upgrade installed packages to newest available versions:
  yum upgrade
zypper

SUSE & openSUSE package management utility.

- Synchronize list of packages and versions available:
  
  `zypper refresh`

- Install a new package:
  
  `zypper install package`

- Remove a package:
  
  `zypper remove package`

- Upgrade installed packages to newest available versions:
  
  `zypper update`

- Search package via keyword:
  
  `zypper search keyword`
3 OSX

airport

Wireless network configuration utility.
- Show current wireless status information:
  airport -I
- Sniff wireless traffic on channel 1:
  airport sniff 1
- Scan for available wireless networks:
  airport -s
- Disassociate from current airport network:
  sudo airport -z

archey

Simple tool for stylishly displaying system information.
- Show system information:
  archey
- Show system information without colored output:
  archey --nocolor
- Show system information, using MacPorts instead of Homebrew:
  archey --macports
- Show system information without IP address check:
  archey --offline
base64

Encode and decode using Base64 representation.

- Encode a file:
  
  base64 -i plain_file

- Decode a file:
  
  base64 -D -i base64_file

- Encode from stdin:
  
  echo plain_text | base64

- Decode from stdin:
  
  echo base64_text | base64 -D

brew

Package manager for OS X.

- Search formula:
  
  brew search text

- Install formula:
  
  brew install formula

- List all installed formulae:
  
  brew list

- Get latest version of installed formula (passing no formula updates all installed formulae):
  
  brew upgrade formula

- Update brew:
  
  brew update

- Display information about formula, which contains formula version, installed path, dependencies, etc.:
  
  brew info formula

- Check your system for potential problems:
  
  brew doctor
caffeinate

Prevent a system from sleeping.
- Prevent mac from sleeping for 1 hour (3600 seconds):
  
  caffeinate -u -t 3600

- Prevent mac from sleeping until a command completes:

  caffeinate -s command

carthage

Carthage is a dependency management tool for Cocoa applications.
- Download and build all dependencies mentioned in Cartfile. Also used to update dependencies to their latest version:

  carthage update

- Update dependencies and only build for iOS:

  carthage update --platform ios

- Update dependencies but don’t build:

  carthage update --no-build

- Download and rebuild the current dependency set without updating:

  carthage bootstrap

- Rebuild a specific dependency:

  carthage build dependency
date

Set or display the system date.
- Display the current date using the default locale’s format:

  date +"%c"

- Display the current date in UTC and ISO 8601 format:

  date -u +"%Y-%m-%dT%H:%M:%SZ"

- Display the current date as a Unix timestamp (seconds since the Unix epoch):

  date +%s

- Display a specific date (represented as a Unix timestamp) using the default format:

  date -r 1473305798
**dd**

Convert and copy a file.

- Make a bootable usb drive from an isohybrid file (such like archlinux-xxx.iso):

  ```
  dd if=file.iso of=/dev/usb_drive
  ```

- Clone a drive to another drive with 4MB block and ignore error:

  ```
  dd if=/dev/source_drive of=/dev/dest_drive bs=4m conv=noerror
  ```

- Generate a file of 100 random bytes by using kernel random driver:

  ```
  dd if=/dev/urandom of=random_file bs=100 count=1
  ```

- Benchmark the write performance of a disk:

  ```
  dd if=/dev/zero of=file_1GB bs=1024 count=1000000
  ```

**defaults**

Read and write OS X user configuration for applications.

- Read system defaults for an application option:

  ```
  defaults read application option
  ```

- Read default values for an application option:

  ```
  defaults read -app application option
  ```

- Write the default value of an application option:

  ```
  defaults write application option -type value
  ```

- Speed up Mission Control animations:

  ```
  defaults write com.apple.Dock expose-animation-duration -float 0.1
  ```

- Delete all defaults of an application:

  ```
  defaults delete application
  ```
**diskutil**

Utility to manage local disks and volumes.

- List all currently available disks, partitions and mounted volumes:
  ```
  diskutil list
  ```

- Repair the file system data structures of a volume:
  ```
  diskutil repairVolume /dev/diskX
  ```

- Unmount a volume:
  ```
  diskutil unmountDisk /dev/diskX
  ```

- Eject a CD/DVD (unmount first):
  ```
  diskutil eject /dev/disk1
  ```

**ditto**

Copy files and folders.

- Overwrite contents of destination folder with contents of source folder:
  ```
  ditto path/to/source path/to/destination
  ```

- Print a line to the Terminal window for every file that’s being copied:
  ```
  ditto -V path/to/source path/to/destination
  ```

- Copy a given file or folder, while retaining the original file permissions:
  ```
  ditto -rsr path/to/source path/to/destination
  ```

**drutil**

Interact with DVD burners.

- Eject a disk from the drive:
  ```
  drutil eject
  ```

- Burn a folder as an ISO9660 filesystem onto a DVD. Don’t verify and eject when complete:
  ```
  drutil burn -noverify -eject -iso9660
  ```
**du**

Disk usage: estimate and summarize file and folder space usage.

- List the sizes of a folder and any subfolders, in the given unit (KB/MB/GB):
  
  `du -k|m|g path/to/folder`

- List the sizes of a folder and any subfolders, in human-readable form (i.e. auto-selecting the appropriate unit for each size):
  
  `du -h path/to/folder`

- Show the size of a single folder, in human readable units:
  
  `du -sh path/to/folder`

- List the human-readable sizes of a folder and of all the files and folders within it:
  
  `du -ah path/to/folder`

- List the human-readable sizes of a folder and any subfolders, up to N levels deep:
  
  `du -h -d N path/to/folder`

- List the human-readable size of all .jpg files in subfolders of the current folder, and show a cumulative total at the end:
  
  `du -ch *//*.jpg`

**head**

Output the first part of files.

- Output the first few lines of a file:
  
  `head -n count_of_lines filename`

- Output the first few bytes of a file:
  
  `head -c number_in_bytes filename`
hostname

Show or set the system’s host name.
- Show current host name:
  hostname
- Set current host name:
  hostname new_hostname

launchctl

A command-line interface to Apple’s launchd manager for launch daemons (system-wide services) and launch agents (per-user programs). launchd loads XML-based *.plist files placed in the appropriate locations, and runs the corresponding commands according to their defined schedule.
- Activate a user-specific agent to be loaded into launchd whenever the user logs in:
  launchctl load ~/Library/LaunchAgents/my_script.plist
- Activate an agent which requires root privileges to run and/or should be loaded whenever any user logs in (note the absence of ~ in the path):
  sudo launchctl load /Library/LaunchAgents/root_script.plist
- Activate a system-wide daemon to be loaded whenever the system boots up (even if no user logs in):
  sudo launchctl load /Library/LaunchDaemons/system_daemon.plist
- Show all loaded agents/daemons, with the PID if the process they specify is currently running, and the exit code returned the last time they ran:
  launchctl list
- Unload a currently loaded agent, e.g. to make changes (note: the plist file is automatically loaded into launchd after a reboot and/or logging in):
  launchctl unload ~/Library/LaunchAgents/my_script.plist
- Manually run a known (loaded) agent/daemon, even if it isn’t the right time (note: this command uses the agent’s label, rather than the filename):
  launchctl start my_script
- Manually kill the process associated with a known agent/daemon, if it’s running:
  launchctl stop my_script
locate

Find filenames quickly.

- Look for pattern in the database. Note: the database is recomputed periodically (usually weekly or daily):

  locate pattern

- Look for a file by its exact filename (a pattern containing no globbing characters is interpreted as *pattern*):

  locate */filename

- Recompute the database. You need to do it if you want to find recently added files:

  sudo /usr/libexec/locate.updatedb

logger

Add messages to syslog (/var/log/syslog).

- Log a message to syslog:

  logger message

- Take input from stdin and log to syslog:

  echo log_entry | logger

- Send the output to a remote syslog server running at a given port. Default port is 514:

  echo log_entry | logger -h hostname -P port

- Use a specific tag for every line logged. Default is the name of logged in user:

  echo log_entry | logger -t tag

- Log messages with a given priority. Default is user.notice. See man logger for all priority options:

  echo log_entry | logger -p user.warning
**look**

Look for lines in sorted file.
- Look for lines which begins with the given prefix:
  
  `look prefix file`
- Look for lines ignoring case:
  
  `look -f prefix file`

**md5**

Calculate MD5 cryptographic checksums.
- Calculate the MD5 checksum for a file:
  
  `md5 filename`
- Calculate MD5 checksums for multiple files:
  
  `md5 filename1 filename2`
- Output only the md5 checksum (no filename):
  
  `md5 -q filename`
- Print a checksum of the given string:
  
  `md5 -s string`

**mdfind**

List files matching a given query.
- Find a file by its name:
  
  `mdfind -name file`
- Find a file by its content:
  
  `mdfind query`
- Find a file containing a string, in a given directory:
  
  `mdfind -onlyin directory query`
**netstat**

Displays various networks related information such as open connections, open socket ports etc.

- List all ports:
  ```bash
  netstat -a
  ```

- List all listening ports:
  ```bash
  netstat -l
  ```

- List listening TCP ports:
  ```bash
  netstat -t
  ```

- Display PID and program names for a specific port:
  ```bash
  netstat -p {PROTOCOL}
  ```

- List information continuously:
  ```bash
  netstat -c
  ```

**networksetup**

Configuration tool for Network System Preferences.

- List available network service providers (Ethernet, Wi-Fi, Bluetooth, etc):
  ```bash
  networksetup -listallnetworkservices
  ```

- Show network settings for a particular networking device:
  ```bash
  networksetup -getinfo "Wi-Fi"
  ```

- Get currently connected Wi-Fi network name (Wi-Fi device usually en0 or en1):
  ```bash
  networksetup -getairportnetwork en0
  ```

- Connect to a particular Wi-Fi network:
  ```bash
  networksetup -setairportnetwork en0 "Airport Network SSID" password
  ```
**nm**

List symbol names in object files (see c++filt).

- List global (extern) functions in a file (prefixed with T):
  
  `nm -g file.o`

- List only undefined symbols in a file:
  
  `nm -u file.o`

- List all symbols, even debugging symbols:
  
  `nm -a file.o`

**open**

Opens files, directories and applications.

- Open a file with the associated application:
  
  `open file.ext`

- Run a graphical MacOSX application:
  
  `open /Applications/Application.app`

- Open the current directory in Finder:
  
  `open .`

- Reveal a file in finder:
  
  `open -R path/to/file`

- Open all the files of a given extension in the current directory with the associated application:
  
  `open *.ext`

**pbcopy**

Place standard output in the clipboard.

- Place the contents of a file in the clipboard:
  
  `pbcopy < file`

- Place the results of a command in the clipboard:
  
  `find . -type t -name "*.png" | pbcopy`
**pbpaste**

Send the contents of the clipboard to standard output.

- Write the contents of the clipboard to a file:
  
  `pbpaste > file`

- Use the contents of the clipboard as input to a command:
  
  `pbpaste | grep foo`

**pmset**

Configure macOS power management settings, as one might do in System Preferences > Energy Saver. Commands that modify settings must begin with `sudo`.

- Display the current power management settings:
  
  `pmset -g`

- Display the current power source and battery levels:
  
  `pmset -g batt`

- Set display to never sleep when on charger power:
  
  `sudo pmset -c displaysleep 0`

- Set display to sleep after 15 minutes when on battery power:
  
  `sudo pmset -b displaysleep 15`

- Schedule computer to automatically wake up every weekday at 9 AM:
  
  `sudo pmset repeat wake MTWRF 09:00:00`

- Restore to system defaults:
  
  `sudo pmset -a displaysleep 10 disksleep 10 sleep 30 womp 1`
**pod**

Dependency manager for Swift and Objective-C Cocoa projects.

- Create a Podfile for the current project with the default contents:
  
  `pod init`

- Download and install all pods defined in the Podfile (that haven’t been installed before):

  `pod install`

- List all available pods:

  `pod list`

- Show the outdated pods (of those currently installed):

  `pod outdated`

- Update all currently installed pods to their newest version:

  `pod update`

- Update a specific (previously installed) pod to its newest version:

  `pod update pod_name`

- Remove CocoaPods from a Xcode project:

  `pod deintegrate xcode_project`

**qlmanage**

QuickLook server tool.

- Display QuickLook for one or multiple files:

  `qlmanage -p filename filename2`

- Compute 300px wide PNG thumbnails of all JPEGs in the current directory and put them in a directory:

  `qlmanage *.jpg -t -s 300 path/to/directory`

- Reset Quicklook:

  `qlmanage -r`
route

Manually manipulate the routing tables. Necessitates to be root.

- Add a route to a destination through a gateway:
  
  `sudo route add dest_ip_address gateway_address`

- Add a route to a /24 subnet through a gateway:
  
  `sudo route add subnet_ip_address/24 gateway_address`

- Run in test mode (does not do anything, just print):
  
  `sudo route -t add dest_ip_address/24 gateway_address`

- Remove all routes:
  
  `sudo route flush`

- Delete a specific route:
  
  `sudo route delete dest_ip_address/24`

say

Converts text to speech.

- Say a phrase aloud:
  
  `say "I like to ride my bike."`

- Read a file aloud:
  
  `say -f filename.txt`

- Say a phrase with a custom voice and speech rate:
  
  `say -v voice -r words_per_minute "I'm sorry Dave, I can't let you do that."`

- List the available voices:
  
  `say -v ?`

- Create an audio file of the spoken text:
  
  `say -o filename.aiff "Here's to the Crazy Ones."`
**sed**

Run replacements based on regular expressions.

- Replace the first occurrence of a string in a file, and print the result:
  
  `sed 's/find/replace/' filename`

- Replace all occurrences of an extended regular expression in a file:
  
  `sed -E 's/regex/replace/g' filename`

- Replace all occurrences of a string in a file, overwriting the file (i.e. in-place):
  
  `sed -i '' 's/find/replace/g' filename`

- Replace only on lines matching the line pattern:
  
  `sed '/line_pattern/s/find/replace/' filename`

- Apply multiple find-replace expressions to a file:
  
  `sed -e 's/find/replace/' -e 's/find/replace/' filename`

- Replace separator / by any other character not used in the find or replace patterns, e.g., #:
  
  `sed 's#find#replace#' filename`

**shutdown**

Shutdown and reboot the system.

- Power off (halt) immediately:
  
  `shutdown -h now`

- Sleep immediately:
  
  `shutdown -s now`

- Reboot immediately:
  
  `shutdown -r now`

- Reboot in 5 minutes:
  
  `shutdown -r +5`
**sw_vers**

Print Mac OSX Software versioning information.
- Print OSX Version:
  ```bash
  sw_vers -productVersion
  ```
- Print OSX Build:
  ```bash
  sw_vers -buildVersion
  ```

**sysctl**

Access kernel state information.
- Show all available variables and their values:
  ```bash
  sysctl -a
  ```
- Show Apple model identifier:
  ```bash
  sysctl -n hw.model
  ```
- Show CPU model:
  ```bash
  sysctl -n machdep.cpu.brand_string
  ```
- Show available CPU features (MMX, SSE, SSE2, SSE3, AES, etc):
  ```bash
  sysctl -n machdep.cpu.feature
  ```
- Set a changeable kernel state variable:
  ```bash
  sysctl -w section.tunable=value
  ```

**system_profiler**

Report system hardware and software configuration.
- Display a full system profiler report which can be opened by System Profiler.app:
  ```bash
  system_profiler -xml > MyReport.spx
  ```
- Display a hardware overview (Model, CPU, Memory, Serial, etc):
  ```bash
  system_profiler SPHardwareDataType
  ```
- Print the system serial number:
  ```bash
  system_profiler SPHardwareDataType|grep "Serial Number (system)" | awk '{print $4}''
  ```
systemsetup

Configure System Preferences machine settings.

- Enable remote login (SSH):
  systemsetup -setremotelogin on

- Specify TimeZone, NTP Server and enable network time:
  systemsetup -settimezone US/Pacific -setnetworktimeserver us.pool.ntp.org -setusingnetworktime on

- Make the machine never sleep and automatically restart on power failure or kernel panic:
  systemsetup -setsleep off -setrestartpowerfailure on -setrestartfreeze on

- List valid startup disks:
  systemsetup -liststartupdisks

- Specify a new startup disk:
  systemsetup -setstartupdisk path

top

Display dynamic real-time information about running processes.

- Start top, all options are available in the interface:
  top

- Start top sorting processes by internal memory size (default order - process ID):
  top -o mem

- Start top sorting processes first by CPU, then by running time:
  top -o cpu -O time

- Start top displaying only processes owned by given user:
  top -user user_name

- Get help about interactive commands:
  ?
tree

Show the contents of the current directory as a tree.
- Show files and directories up to 'num' levels of depth (where 1 means the current directory):
  `tree -L num`
- Show directories only:
  `tree -d`
- Show hidden files too:
  `tree -a`
- Print the tree without indentation lines, showing the full path instead (use -N to not escape whitespace and special characters):
  `tree -i -f`
- Print the size of each node next to it, in human-readable format, with folders displaying their cumulative size (as in the du command):
  `tree -s -h --du`
- Find files within the tree hierarchy, using a wildcard (glob) pattern, and pruning out directories that don't contain matching files:
  `tree -P '*.txt' --prune`
- Find directories within the tree hierarchy, pruning out directories that aren't ancestors of the wanted one:
  `tree -P directory_name --matchdirs --prune`

uname

Print details about the current machine and the operating system running on it. Note: for additional information about the operating system, try the `sw_vers` command.
- Print hardware-related information: machine and processor:
  `uname -mp`
- Print software-related information: operating system, release number, and version:
  `uname -srv`
- Print the nodename (hostname) of the system:
  `uname -n`
- Print all available system information (hardware, software, nodename):
  `uname -a`
w

Show who is logged on and what they are doing. Print user login, TTY, remote host, login time, idle time, current process.

- Show logged-in users info:
  
  `w`

- Show logged-in users info without a header:
  
  `w -h`

- Show info about logged-in users, sorted by their idle time:
  
  `w -i`

wacaw

A little command-line tool for Mac OS X that allows you to capture both still pictures and video from an attached camera.

- Take a picture from webcam:
  
  `wacaw filename`

- Record a video:
  
  `wacaw --video filename -D duration_in_seconds`

- Take a picture with custom resolution:
  
  `wacaw -x width -y height filename`

- Copy image just taken to clipboard:
  
  `wacaw --to-clipboard`

- List the devices available:
  
  `wacaw -L`
**xattr**

Utility to work with extended filesystem attributes.

- List key:value extended attributes for a given file:
  
  `xattr -l file`

- Write an attribute for a given file:
  
  `xattr -w attribute_key attribute_value file`

- Delete an attribute from a given file:
  
  `xattr -d attribute_key file`

- Delete all extended attributes from a given file:
  
  `xattr -c file`

- Recursively delete an attribute in a given directory:
  
  `xattr -rd attribute_key directory`

**xcodebuild**

Build Xcode projects.

- Build workspace:
  
  `xcodebuild -workspace workspace_name.workspace -scheme scheme_name -configuration configuration_name clean build SYMROOT=SYMROOT_path`

- Build project:
  
  `xcodebuild -target target_name -configuration configuration_name clean build SYMROOT=SYMROOT_path`

- Show SDKs:
  
  `xcodebuild -showsdks`
**xc_tool**

Tool for building Xcode projects.

- Build a single project without any workspace:
  
  xc_tool -project *YourProject*.xcodeproj -scheme *YourScheme* build

- Build a project that is part of a workspace:
  
  xc_tool -workspace *YourWorkspace*.xcworkspace -scheme *YourScheme* build

- Clean, build and execute all the tests:
  
  xc_tool -workspace *YourWorkspace*.xcworkspace -scheme *YourScheme* clean build test

**xed**

Opens files for editing in XCode.

- Open file in XCode:
  
  xed *file1*

- Open file(s) in XCode, create if it doesn’t exist:
  
  xed -c *filename1*

- Open a file in XCode and jump to line number 75:
  
  xed -l 75 *filename*

**xsltproc**

Transform XML with XSLT to produce output (usually HTML or XML).

- Transform an XML file with a specific XSLT stylesheet:
  
  xsltproc --output *output.html* stylesheet.xslt *xmlfile.xml*

- Pass a value to a parameter in the stylesheet:
  
  xsltproc --output *output.html* --stringparam name value stylesheet.xslt *xmlfile.xml*
4 SUNOS

devfsadm

Administration command for /dev. Maintains the /dev namespace.

- Scan for new disks:
  
  devfsadm -c disk

- Cleanup any dangling /dev links and scan for new device:
  
  devfsadm -C -v

- Dry-run - output what would be changed but make no modifications:
  
  devfsadm -C -v -n

prctl

Get or set the resource controls of running processes, tasks, and projects.

- Examine process limits and permissions:
  
  prctl PID

- Examine process limits and permissions in machine parseable format:
  
  prctl -P PID

- Get specific limit for a running process:
  
  prctl -n process.max-file-descriptor PID
prstat
Report active process statistics.
- Examine all processes and reports statistics sorted by CPU usage:
  prstat
- Examine all processes and reports statistics sorted by memory usage:
  prstat -s rss
- Report total usage summary for each user:
  prstat -t
- Report microstate process accounting information:
  prstat -m
- Print out a list of top 5 cpu using processes every second:
  prstat -c -n 5 -s cpu 1

svcadm
Manipulate service instances.
- Enable a service in the service database:
  svcadm enable service_name
- Disable service:
  svcadm disable service_name
- Restart a running service:
  svcadm restart service_name
- Command service to re-read configuration files:
  svcadm refresh service_name
- Clear a service from maintenance state and command it to start:
  svcadm clear service_name
**svccfg**

Import, export, and modify service configurations.

- Validate configuration file:
  
  `svccfg validate smf.xml`

- Export service configurations to file:
  
  `svccfg export servicename > smf.xml`

- Import/update service configurations from file:
  
  `svccfg import smf.xml`

**SVCS**

List information about running services.

- List all running services:
  
  `svcs`

- List services that are not running:
  
  `svcs -vx`

- List information about a service:
  
  `svcs apache`

- Show location of service log file:
  
  `svcs -L apache`

- Display end of a service log file:
  
  `tail $(svcs -L apache)`