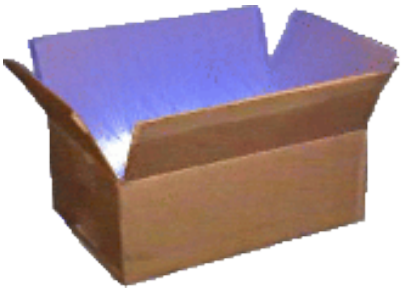
Busybox

- [Busybox] Cheat-sheet

[Busybox] Cheat-sheet

Introduction

Busybox est une boîte à outils qui embarque beaucoup de binaires basiques et très pratiques pour dépanner ou effectuer tout un tas de tâches. Il peut aussi très bien être détourné pour être utilisé à des fins malveillantes.



Cheat-sheet

Aide

Le manuel est très complet :

```
busybox --help
```

```
BusyBox v1.35.0 (Debian 1:1.35.0-4+b5) multi-call binary.
BusyBox is copyrighted by many authors between 1998-2015.
Licensed under GPLv2. See source distribution for detailed
copyright notices.

Usage: busybox [function [arguments]...]
   or: busybox --list[-full]
   or: busybox --install [-s] [DIR]
   or: function [arguments]...

        BusyBox is a multi-call binary that combines many common Unix
        utilities into a single executable.  The shell in this build
```

```
        is configured to run built-in utilities without $PATH search.
        You don't need to install a link to busybox for each utility.
        To run external program, use full path (/sbin/ip instead of ip).

Currently defined functions:
        [, [[, acpid, adjtimex, ar, arch, arp, arping, ascii, ash, awk, base64, basename, bc,
blkdiscard, blockdev, brctl, bunzip2, busybox, bzcat, bzip2, cal, cat, chgrp, chmod, chown,
chroot, chvt, clear, cmp,
        cp, cpio, crc32, crond, crontab, cttyhack, cut, date, dc, dd, deallocvt, depmod,
devmem, df, diff, dirname, dmesg, dnsdomainname, dos2unix, dpkg, dpkg-deb, du, dumpkmap,
dumpleases, echo, ed, egrep, env,
        expand, expr, factor, fallocate, false, fatattr, fdisk, fgrep, find, findfs, fold,
free, freeramdisk, fsfreeze, fstrim, ftpget, ftpput, getopt, getty, grep, groups, gunzip,
gzip, halt, head, hexdump,
        hostid, hostname, httpd, hwclock, i2cdetect, i2cdump, i2cget, i2cset, i2ctransfer, id,
ifconfig, ifdown, ifup, init, insmod, ionice, ip, ipcalc, ipneigh, kill, killall, klogd, last,
less, link, linux32,
        linux64, linuxrc, ln, loadfont, loadkmap, logger, login, logname, logread, losetup,
ls, lsmod, lsscsi, lzcat, lzma, lzop, md5sum, mdev, microcom, mim, mkdir, mkdosfs, mke2fs,
mkfifo, mknod, mkpasswd,
        mkswap, mktemp, modinfo, modprobe, more, mount, mt, mv, nameif, nc, netstat, nl,
nologin, nproc, nsenter, nslookup, nuke, od, openvt, partprobe, passwd, paste, patch, pidof,
ping, ping6, pivot_root,
        poweroff, printf, ps, pwd, rdate, readlink, realpath, reboot, renice, reset, resume,
rev, rm, rmdir, rmmod, route, rpm, rpm2cpio, run-init, run-parts, sed, seq, setkeycodes,
setpriv, setsid, sh, sha1sum,
        sha256sum, sha3sum, sha512sum, shred, shuf, sleep, sort, ssl_client, start-stop-
daemon, stat, strings, stty, su, sulogin, svc, svok, swapoff, swapon, switch_root, sync,
sysctl, syslogd, tac, tail, tar,
        taskset, tc, tee, telnet, telnetd, test, tftp, time, timeout, top, touch, tr,
traceroute, traceroute6, true, truncate, ts, tty, tunctl, ubirename, udhcpc, udhcpd, uevent,
umount, uname, uncompress,
        unexpand, uniq, unix2dos, unlink, unlzma, unshare, unxz, unzip, uptime, usleep,
uudecode, uuencode, vconfig, vi, w, watch, watchdog, wc, wget, which, who, whoami, xargs, xxd,
xz, xzcat, yes, zcat
```

Shell busybox

Pour entrer dans le shell interractif busybox :

```
busybox sh
```

> Cela est plus pratique si vous avez beaucoup d'applets à utiliser plutôt que de taper busybox <function> à chaque fois vous pourrez directement appeler les fonctions dans ce shell.

Serveur web

Démarrer un serveur web dans le dossier courant :

```
busybox httpd -vv -f -p 8080
```

Encodage ASCII

Pour encoder en ASCII :

```
printf <STRING> | busybox uuencode - |  sed -n '2p'
```

Pour décoder en ASCII :

```
s="<ASCII_STR>";printf 'begin 644 -\n%s\n`\nend\n' "$s" | busybox uudecode -o /dev/stdout
```

Encodage base64

Pour encoder en base64 :

```
printf "<STR>" | uuencode -m - | sed -n '2p'
```

Pour décoder en base64 :

```
s="<B64_STR>";printf 'begin-base64 644 -\n%s\n`\n====\n' "$s" | busybox uudecode -o
/dev/stdout
```

Découverte du réseau

Pour découvrir les hôtes actifs sur le réseau avec un scan ARP :

```
for i in $(seq 1 254); do sudo arping -c 1 -w 0 -I wlp0s20f3 192.168.1.$i >/dev/null 2>&1 &&
echo "192.168.1.$i"; done
```

> Ce n'est pas aussi rapide qu'un nmap mais sûrement beaucoup plus discret.