The essential tools for ease of management

GAIA EDITION
Updated for R77.30

Smart system administration with

UNItools

Compiled by Niels Thomas Haugård

This package is the pre-requisite for packages like UNlha, UNlfw1lr and UNlfw1doc

UNItools Table of Contents

Package	Version		Description
lshw	B.02.16	Name Vendor URL	HardWare LiSter for Linux Lyonel Vincent <lyonel@ezix.org> http://ezix.org/software/files/lshw-B.02.16.tar.gz
iftop	0.16	Name Vendor URL	iftop http://www.ex-parrot.com/~pdw/iftop/download/iftop-0.16.tar.gz http://www.ex-parrot.com/~pdw/iftop/download/iftop-0.16.tar.gz
iptraf	2.7.0	Name Vendor URL	iptraf ftp://iptraf.seul.org/pub/iptraf/iptraf-2.7.0.tar.gz ftp://iptraf.seul.org/pub/iptraf/iptraf-2.7.0.tar.gz
minicom	2.1	Name Vendor URL	minicom http://alioth.debian.org/download.php/123/minicom-2.1.tar.gz http://alioth.debian.org/download.php/123/minicom-2.1.tar.gz
lrzsz	0.12.20	Name Vendor URL	XMODEM, YMODEM, ZMODEM (Batch) file send and receive Uwe Ohse http://freshmeat.net/redir/lrzsz/6021/url_tgz/lrzsz-0.12.20.tar.gz
ofiller	2.0	Name Vendor URL	Object Filler and Object Dumper for Check Point Martin Hoz, (c) 2003-2005 by Check Point Software Technologies, Ltd. and subsidiaries http://www.lindercentral.com/ofiller/ofiller_2.0.tgz
lft	3.36	Name Vendor URL	lft http://pwhois.org/ http://pwhois.org/
di	4.9	Name Vendor URL	disk information utility Brad Lanam, Walnut Creek, CA, USA http://www.gentoo.com/di/di-4.9.tar.gz
atop	1.19	Name Vendor URL	AT Computing's System & Process Monitor Gerlof Langeveld, AT Computing, Nijmegen, The Netherlands, gerlof@ATComputing.nl http://www.atconsultancy.nl/atop/packages/atop-1.19.tar.gz
dialog	1.0-20051107	Name Vendor URL	display dialog boxes from shell scripts - T.Dickey <dickey@invisible-island.net> http://invisible-island.net/dialog
conf_template	1.0	Name Vendor URL	SPLAT dot-profiles and config files for the smtp client I2 http://cppkg.haugaard.net/conf_template.1.0
miniweb	208	Name Vendor URL	miniweb https://sourceforge.net/code-snapshots/svn/m/mi/miniweb/code/miniweb-code-208.zip https://sourceforge.net/code-snapshots/svn/m/mi/miniweb/code/miniweb-code-208.zip
smtpclient	1.0.0	Name Vendor URL	smtpclient: simple SMTP client http://www.engelschall.com/sw/smtpclient ftp://www.ossp.org/pkg/tool/smtpclient/smtpclient-1.0.0.tar.gz
lighthttpd	1.4.39	Name Vendor URL	lighthttpd http://download.lighttpd.net/lighttpd/releases-1.4.x/lighttpd-1.4.39.tar.gz http://download.lighttpd.net/lighttpd/releases-1.4.x/lighttpd-1.4.39.tar.gz
tcptrack	1.1.4	Name Vendor	Monitor TCP connections on the network Steve Benson <steve@rhythm.cx> Leo Costela <costela@debian.org>

		URL	http://freshmeat.net/redir/tcptrack/45771/url_tgz/tcp-track-1.1.5.tar.gz
libsmbios	2.2.19	Name Vendor URL	Dell hardware tools Dell Computer Corporation http://linux.dell.com/libsmbios/download/libsmbios/libsm-bios-2.2.19/libsmbios-2.2.19.tar.gz
strings	2.14	Name Vendor URL	print the strings of printable characters in files GNU ftp://ftp.gnu.org/gnu/binutils/binutils-2.14.tar.gz
rcs	5.8	Name Vendor URL	Revision Control System GNU - Walter F. Tichy http://mirrors.dotsrc.org/gnu/rcs/rcs-5.8.tar.gz
p0f	3.06b	Name Vendor URL	passive OS fingerprinting tool http://lcamtuf.coredump.cx/p0f3/releases/p0f-3.06b.tgz http://lcamtuf.coredump.cx/p0f3/releases/p0f-3.06b.tgz
nmap	3.93	Name Vendor URL	Network exploration tool and security scanner Fyodor <fyodor@insecure.org> http://www.insecure.org/nmap/dist/nmap-3.93.tgz
realpath	1.0	Name Vendor URL	Return the real path of a link Niels Thomas Haugaard http://cpfw_pkgs.haugaard.net
unbound_dns	1.4.22	Name Vendor URL	unbound NLnet Labs http://unbound.net/downloads/unbound-1.4.22.tar.gz
fping	3.9	Name Vendor URL	fping http://fping.org/dist/fping-3.9.tar.gz http://fping.org/dist/fping-3.9.tar.gz
scripts	1.0	Name Vendor URL	Special purpose monitors and system utilities i2 - mostly http://apollon.ssi.uni-c.dk
wget	1.10.2	Name Vendor URL	non-interactive network downloader see AUTHORS http://freshmeat.net/redir/wget/3569/url_tgz/wget-1.10.2.tar.gz
screen	4.0.3	Name Vendor URL	screen gnu http://ftp.gnu.org/gnu/screen/screen-4.0.3.tar.gz
memtester	4.3.0	Name Vendor URL	memtester http://pyropus.ca/software/memtester/old-ver-sions/memtester-4.3.0.tar.gz http://pyropus.ca/software/memtester/old-ver-sions/memtester-4.3.0.tar.gz
pktstat	1.8.5	Name Vendor URL	display packet activity on a crt David Leonard, leonard@users.sourceforge.net http://www.adaptive-enterprises.com.au/~d/software/pktstat/pkt-stat-1.8.5.tar.gz
fwbu	1.0	Name Vendor URL	Check Point firewall-1 backup Niels Thomas Haugård internal
nipper	0.11.5	Name Vendor URL	nipper http://www.darknet.org.uk/content/files/nipper-0.11.5.tgz http://www.darknet.org.uk/content/files/nipper-0.11.5.tgz

lft	2.5	Name Vendor URL	Alternative traceroute tool for network (reverse) engineers Victor Oppleman, Eugene Antsilevitch http://oppleman.com/lft/lft-2.5.tar.gz
host	991529	Name Vendor URL	DNS lookup utility Eric Wassenaar http://www.nikhef.nl/user/e07/host_991529.tar.gz
dmidecode	2.12	Name Vendor URL	dmidecode http://download.savannah.gnu.org/releases/dmidecode/dmide-code-2.12.tar.gz http://download.savannah.gnu.org/releases/dmidecode/dmide-code-2.12.tar.gz
tcpstat	1.5	Name Vendor URL	tcpstat http://www.frenchfries.net/paul/tcpstat/tcpstat-1.5.tar.gz http://www.frenchfries.net/paul/tcpstat/tcpstat-1.5.tar.gz
setup	1.0	Name Vendor URL	dims os selv intern
rsync	2.6.6	Name Vendor URL	faster, flexible replacement for scp Martin Pool <mbp@samba.org> http://rsync.samba.org/ftp/rsync/rsync-2.6.6.tar.gz
links	1.00pre14	Name Vendor URL	Text WWW browser Mikulas Patocka <mikulas@artax.karlin.mff.cuni.cz> and others. See AUTHORS http://artax.karlin.mff.cuni.cz/~mikulas/links/down-load/links-1.00pre14.tar.gz
ifstatus	1.1.0	Name Vendor URL	ifstatus http://ifstatus.sourceforge.net/download/ifstatus-v1.1.0.tar.gz http://ifstatus.sourceforge.net/download/ifstatus-v1.1.0.tar.gz
saidar	0.14	Name Vendor URL	curses-based tool for viewing system statistics http://www.i-scream.org http://www.mirrorservice.org/sites/ftp.i-scream.org/pub/i-scream/libstatgrab/libstatgrab-0.14.tar.gz
cbm	0.1	Name Vendor URL	cbm http://www.isotton.com/utils/cbm/download/cbm-0.1.tar.gz http://www.isotton.com/utils/cbm/download/cbm-0.1.tar.gz
dhcpdmon	1.2	Name Vendor URL	Monitor and restart dhcpdaemon upon faileures Niels Thomas Haugaard http://www.mavetju.org/download/dhcping-1.2.tar.gz
nload	0.7.4	Name Vendor URL	nload Copyright (C) 2001 - 2011 by Roland Riegel <feedback@roland-riegel.de> http://www.roland-riegel.de/nload/nload-0.7.4.tar.gz
strace	4.5.12	Name Vendor URL	strace http://switch.dl.sourceforge.net/source-forge/strace/strace-4.5.12.tar.bz2 http://switch.dl.sourceforge.net/source-forge/strace/strace-4.5.12.tar.bz2
mtr	0.69	Name Vendor URL	a network diagnostic tool Matt Kimball <mkimball@xmission.com> - see AUTHORS for a full list ftp://ftp.bitwizard.nl/mtr/mtr-0.69.tar.gz

unbound_dns	1.4.20	Name Vendor URL	unbounddns NLnet Labs http://unbound.net/downloads/unbound-1.4.20.tar.gz
bmon	2.0.1	Name Vendor URL	Portable bandwidth monitor and rate estimator Thomas Graf <tgraf@suug.ch> http://freshmeat.net/redir/bmon/36580/url_tgz/bmon-2.0.1.tar.gz
itop	0.1.1	Name Vendor URL	top-like interrupt load monitor Benedikt <hunz@hunz.org> (http://hunz.org) http://freshmeat.net/redir/itop/17843/url_bz2/itop-0.1.tar.bz2
openssl	1.0.2d	Name Vendor URL	openssl https://www.openssl.org/source/openssl-1.0.2d.tar.gz https://www.openssl.org/source/openssl-1.0.2d.tar.gz
postfix	3.0.2	Name Vendor URL	postfix http://mirror.netinch.com/pub/postfix/postfix-release/official/postfix-3.0.2.tar.gz http://mirror.netinch.com/pub/postfix/postfix-release/official/postfix-3.0.2.tar.gz
hping	2.0.0-rc3	Name Vendor URL	network tool able to send custom ICMP/UDP/TCP packets antirez@invece.org http://www.hping.org/hping2.0.0-rc3.tar.gz
perl	5.8.7.815	Name Vendor URL	Practical Extraction and Report Language Active State http://ftp.activestate.com/ActivePerl/Linux/5.8/ActivePerl-5.8.7.815-i686-linux-2.2.17-gcc-211909.tar.gz
dnstop	20121017	Name Vendor URL	dnstop http://dns.measurement-factory.com/tools/dnstop/src/dnstop-20121017.tar.gz http://dns.measurement-factory.com/tools/dnstop/src/dnstop-20121017.tar.gz
netperf	2.4.1	Name Vendor URL	benchmark to measure various aspect of networking performance rick jones - see AUTHORS ftp://ftp.netperf.org/netperf/netperf-2.4.1.tar.gz
edge_backup	1.0	Name Vendor URL	System to backup edge configs to a service center Niels Thomas Haugaard Internally developed

The next pages describes the manual pages for the installed versions of the software components.

NAME

lshw − list hardware

SYNOPSIS

lshw [**-version**]

lshw [**-help**]

lshw [**-X**]

lshw [**[-html] [-short] [-xml] [-json] [-businfo]**] [**-dump** *filename*] [**-class** *class...*] [**-disable** *test...*] [**-enable** *test...*] [**-sanitize**] [**-numeric**] [**-quiet**]

DESCRIPTION

lshw is a small tool to extract detailed information on the hardware configuration of the machine. It can report exact memory configuration, firmware version, mainboard configuration, CPU version and speed, cache configuration, bus speed, etc. on DMI-capable x86 or IA-64 systems and on some PowerPC machines (PowerMac G4 is known to work).

It currently supports DMI (x86 and IA-64 only), OpenFirmware device tree (PowerPC only), PCI/AGP, CPUID (x86), IDE/ATA/ATAPI, PCMCIA (only tested on x86), SCSI and USB.

-version
Displays the version of **lshw** and exits.

-help Displays the available command line options and quits.

-X Launch the X11 GUI (if available).

-html Outputs the device tree as an HTML page.

-xml Outputs the device tree as an XML tree.

-json Outputs the device tree as a JSON object (JavaScript Object Notation).

-short Outputs the device tree showing hardware paths, very much like the output of HP-UX's **ioscan**.

-businfo
Outputs the device list showing bus information, detailing SCSI, USB, IDE and PCI addresses.

-dump *filename*
Dump collected information into a file (SQLite database).

-class *class*
Only show the given class of hardware. *class* can be found using **lshw -short** or **lshw -businfo**.

-C *class*
Alias for **-class** *class*.

-enable *test*

-disable *test*
Enables or disables a test. *test* can be **dmi** (for DMI/SMBIOS extensions), **device-tree** (for OpenFirmware device tree), **spd** (for memory Serial Presence Detect), **memory** (for memory-size guessing heuristics), **cpuinfo** (for kernel-reported CPU detection), **cpuid** (for CPU detection), **pci** (for PCI/AGP access), **isapnp** (for ISA PnP extensions), **pcmcia** (for PCMCIA/PCCARD), **ide** (for IDE/ATAPI), **usb** (for USB devices),**scsi** (for SCSI) or **network** (for network interfaces detection).

-quiet Don't display status.

-sanitize
Remove potentially sensitive information from output (IP addresses, serial numbers, etc.).

-numeric
Also display numeric IDs (for PCI and USB devices).

BUGS

lshw currently does not detect Firewire(IEEE1394) devices.

Not all architectures supported by GNU/Linux are fully supported (e.g. CPU detection).

"Virtual" SCSI interfaces used for SCSI emulation over IDE are not reported correctly yet.

NOTES

lshw must be run as super user or it will only report partial information.

FILES

/usr/local/share/pci.ids

/usr/share/pci.ids

/etc/pci.ids

/usr/share/hwdata/pci.ids
A list of all known PCI ID's (vendors, devices, classes and subclasses).

/proc/bus/pci/*
Used to access the configuration of installed PCI busses and devices.

/proc/ide/*
Used to access the configuration of installed IDE busses and devices.

/proc/scsi/*, /dev/sg*
Used to access the configuration of installed SCSI devices.

/dev/cpu/*/cpuid
Used on x86 platforms to access CPU-specific configuration.

/proc/device-tree/*
Used on PowerPC platforms to access OpenFirmware configuration.

/proc/bus/usb/*
Used to access the configuration of installed USB busses and devices.

/sys/* Used on 2.6 kernels to access hardware/driver configuration information.

EXAMPLES

lshw -short
Lists hardware in a compact format.

lshw -class disk -class storage
Lists all disks and storage controllers in the system.

lshw -html -class network
Lists all network interfaces in HTML.

lshw -disable dmi
Don't use DMI to detect hardware.

SEE ALSO

*/proc/**, **linuxinfo**(1), **lspci**(8), **lsusb**(8)

COPYING

lshw is distributed under the GNU GENERAL PUBLIC LICENSE (GPL) version 2.

AUTHOR

lshw is maintained by Lyonel Vincent <lyonel@ezix.org>.

OTHER INFO

The webpage for **lshw** is at
<URL:http://lshw.org/>

NAME

atop — AT Computing's System & Process Monitor

SYNOPSIS

Interactive usage:

atop [-g|-m|-d|-n|-u|-p|-s|-c|-v] [-C|-M|-D|-N|-A] [-af1x] [*interval* [*samples*]]

Writing and reading raw logfiles:

atop -w *rawfile* [-a] [-S] [*interval* [*samples*]]
atop -r [*rawfile*] [-b *hh:mm*] [-e *hh:mm*] [-g|-m|-d|-n|-u|-p|-s|-c|-v] [-C|-M|-D|-N|-A] [-f1x]

DESCRIPTION

The program *atop* is an interactive monitor to view the load on a Linux system. It shows the occupation of the most critical hardware resources (from a performance point of view) on system level, i.e. cpu, memory, disk and network.
It also shows which processes are responsible for the indicated load with respect to cpu- and memory load on process level; disk- and network load is only shown per process if a kernel patch has been installed.

Every *interval* (default: 10 seconds) information is shown about the resource occupation on system level (cpu, memory, disks and network layers), followed by a list of processes which have been active during the last interval (note that all processes that were unchanged during the last interval are not shown, unless the key 'a' has been pressed). If the list of active processes does not entirely fit on the screen, only the top of the list is shown (sorted in order of activity).
The intervals are repeated till the number of *samples* (specified as command argument) is reached, or till the key 'q' is pressed in interactive mode.

When *atop* is started, it checks whether the standard output channel is connected to a screen, or to a file/pipe. In the first case it produces screen control codes (via the curses library) and behaves interactively; in the second case it produces flat ASCII-output.

In interactive mode, the output of *atop* can be controlled by pressing particular keys. However it is also possible to specify such key as **flag** on the command line. In the latter case *atop* will switch to the indicated mode on beforehand; this mode can be modified again interactively. Specifying such key as flag is especially useful when running *atop* with output to a pipe or file (non-interactively). The flags used are the same as the keys which can be pressed in interactive mode (see section INTERACTIVE COMMANDS).
Additional flags are available to support storage of atop-data in raw format (see section RAW DATA STORAGE).

PROCESS ACCOUNTING

When *atop* is started, it switches on the process accounting mechanism in the kernel. This forces the kernel to write a record with accounting information to the accounting file whenever a process ends. Apart from the kernel administration related to the running processes, *atop* also interprets the accounting records on disk with every interval; in this way *atop* can also show the activity of a process during the interval in which it is finished.
Whenever the last incarnation of *atop* stops (either by pressing 'q' or by 'kill -15'), it switches off the process accounting mechanism again. You should never terminate *atop* by 'kill -9', because then it has no chance to stop process accounting; as a result the accounting file may consume a lot of disk space after a while.

COLORS

For the resource consumption on system level, *atop* uses colors to indicate that a critical occupation percentage has been (almost) reached. A critical occupation percentage means that is likely that this load causes a noticable negative performance influence for applications using this resource. The critical percentage depends on the type of resource: e.g. the performance influence of a disk with a busy percentage of 80% might be more noticable for applications/user than a CPU with a busy percentage of 90%.
Currently *atop* uses the following default values to calculate a weighted percentage per resource:

Processor
A busy percentage of 90% or higher is considered ‘critical’.

Disk
A busy percentage of 70% or higher is considered ‘critical’.

Network
A busy percentage of 90% or higher for the load of an interface is considered ‘critical’.

Memory
An occupation percentage of 90% is considered ‘critical’. Notice that this occupation percentage is the accumulated memory consumption of the kernel (including slab) and all processes; the memory for the page cache (‘cache’ and ‘buff’ in the MEM-line) is not implied!
If the number of pages swapped out (‘swout’ in the PAG-line) is larger than 10 per second, the memory resource is considered ‘critical’.
If the committed virtual memory exceeds the limit (‘vmcom’ and ‘vmlim’ in the SWP-line), the SWP-line is colored due to overcommitting the system.

Swap
An occupation percentage of 80% is considered ‘critical’ because swap space might be completely exhausted in the near future; it is not critical from a performance point-of-view.

These default values can be modified in the configuration file (see section CONFIGURATION FILE).

When a resource exceeded its critical occupation percentage, the entire screen line is colored red.
When a resource exceeded (default) 80% of its critical percentage (so it is almost critical), the entire screen line is colored cyan. This ‘almost critical percentage’ (one value for all resources) can be modified in the configuration file (see section CONFIGURATION FILE).

With the key ’x’ (or flag -x), line coloring can be suppressed.

INTERACTIVE COMMANDS

When running *atop* interactively (no output redirection), keys can be pressed to control the output. In general, lower case keys can be used to show other information for the active processes and upper case keys can be used to influence the sort order of the active process list.

g Show generic output (default).

Per process the following fields are shown: process-id, cpu consumption during the last interval in system- and user mode, the virtual and resident memory growth of the process.
The subsequent columns contain the username, number of threads in the thread group, the status and exit code. However if the kernel patch ‘cnt’ has been installed, the number of read- and write transfers, and the number of received and sent network packets are shown.
The last columns contain the state, the occupation percentage for the choosen resource (default: cpu) and the process name.

m Show memory related output.

Per process the following fields are shown: process-id, minor and major memory faults, size of virtual shared text, total virtual process size, total resident process size, virtual and resident growth during last interval, memory occupation percentage and process name.

d Show disk-related output.

Per process the following fields are shown: process-id, number of physical disk reads, average size per read (bytes), total size for read transfers, physical disk writes, average size per write (bytes), total size for write transfers, disk occupation percentage and process name.
This information can only be shown when kernel patch ‘cnt’ is installed.

n Show network related output.

Per process the following fields are shown: process-id, number of received TCP packets with the average size per packet (in bytes), number of sent TCP packets with the average size per packet (in bytes), number of received UDP packets with the average size per packet (in bytes), number of sent UDP packets with the average size per packet (in bytes), and received and send raw packets (e.g. ICMP) in one column, the network occupation percentage and process name.
This information can only be shown when kernel patch 'cnt' is installed.

s Show scheduling characteristics.

Per process/thread the following fields are shown: process-id, thread group id, number of threads in thread group, scheduling policy (normal timesharing, realtime round-robin, realtime fifo), nice value, priority, realtime priority, current processor, status, state, the occupation percentage for the choosen resource and the process name.

v Show various process characteristics.

Per process the following fields are shown: process-id, user name and group, start date and time, status (e.g. exit code if the process has finished), state, the occupation percentage for the choosen resource and the process name.

c Show the command line of the process.

Per process the following fields are shown: process-id, the occupation percentage for the choosen resource and the command line including arguments.

u Show the process activity accumulated per user.

Per user the following fields are shown: number of processes active or terminated during last interval (or in total if combined with command 'a'), accumulated cpu consumption during last interval in system- and user mode, the current virtual and resident memory space consumed by active processes (or all processes of the user if combined with command 'a').
When the kernel patch 'cnt' has been installed, the accumulated number of read- and write transfers on disk, and the number of received and sent network packets are shown. When the kernel patch is not installed, these counters are zero.
The last columns contain the accumulated occupation percentage for the choosen resource (default: cpu) and the user name.

p Show the process activity accumulated per program (i.e. process name).

Per program the following fields are shown: number of processes active or terminated during last interval (or in total if combined with command 'a'), accumulated cpu consumption during last interval in system- and user mode, the current virtual and resident memory space consumed by active processes (or all processes of the user if combined with command 'a').
When the kernel patch 'cnt' has been installed, the accumulated number of read- and write transfers on disk, and the number of received and sent network packets are shown. When the kernel patch is not installed, these counters are zero.
The last columns contain the accumulated occupation percentage for the choosen resource (default: cpu) and the program name.

C Sort the current list in the order of cpu consumption (default). The one-but-last column changes to ''CPU''.

M Sort the current list in the order of resident memory consumption. The one-but-last column changes to ''MEM''.

D Sort the current list in the order of disk accesses issued. The one-but-last column changes to ‘‘DSK’’.

N Sort the current list in the order of network packets received/transmitted. The one-but-last column changes to ‘‘NET’’.

A Sort the current list automatically in the order of the most busy system resource during this interval. The one-but-last column shows either ‘‘ACPU’’, ‘‘AMEM’’, ‘‘ADSK’’ or ‘‘ANET’’ (the preceding ’A’ indicates automatic sorting-order). The most busy resource is determined by comparing the weighted busy-percentages of the system resources, as described earlier in the section COLORS.
This option remains valid until another sorting-order is explicitly selected again.
A sorting-order for disk or network is only possible when kernel patch ‘cnt’ is installed.

Miscellaneous interactive commands:

? Request for help information (also the key ’h’ can be pressed).

V Request for version information (version number and date).

x Use colors to highlight critical resources (toggle).

z The pause key can be used to freeze the current situation in order to investigate the output on the screen. While *atop* is paused, the keys described above can be pressed to show other information about the current list of processes. Whenever the pause key is pressed again, atop will continue with a next sample.

i Modify the interval timer (default: 10 seconds). If an interval timer of 0 is entered, the interval timer is switched off. In that case a new sample can only be triggered manually by pressing the key ’t’.

t Trigger a new sample manually. This key can be pressed if the current sample should be finished before the timer has exceeded, or if no timer is set at all (interval timer defined as 0). In the latter case *atop* can be used as a stopwatch to measure the load being caused by a particular application transaction, without knowing on beforehand how many seconds this transaction will last.

When viewing the contents of a raw file, this key can be used to show the next sample from the file.

T When viewing the contents of a raw file, this key can be used to show the previous sample from the file.

r Reset all counters to zero to see the system and process activity since boot again.

When viewing the contents of a raw file, this key can be used to rewind to the beginning of the file again.

U Specify a search string for specific user names as a regular expression. From now on, only (active) processes will be shown from a user which matches the regular expression. The system statistics are still system wide. If the Enter-key is pressed without specifying a name, active processes of all users will be shown again.

P Specify a search string for specific process names as a regular expression. From now on, only processes will be shown with a name which matches the regular expression. The system statistics are still system wide. If the Enter-key is pressed without specifying a name, all active processes will be shown again.

a The ‘all/active’ key can be used to toggle between only showing/accumulating the processes that were active during the last interval (default) or showing/accumulating all processes.

f Fixate the number of lines for system resources (toggle). By default only the lines are shown about system resources (cpu, paging, disk, network) that really have been active during the last interval.

With this key you can force *atop* to show lines of inactive resources as well.

1 Show relevant counters as an average per second (in the format '..../s') instead of as a total during the interval (toggle).

l Limit the number of system level lines for the counters per-cpu, the active disks and the network interfaces. By default lines are shown of all cpu's, disks and network interfaces which have been active during the last interval. Limiting these lines can be useful on systems with huge number cpu's, disks or interfaces in order to be able to run *atop* on a screen/window with e.g. only 24 lines.
For all mentioned resources the maximum number of lines can be specified interactively. When using the flag **-l** the maximum number of per-cpu lines is set to 0, the maximum number of disk lines to 5 and the maximum number of interface lines to 3. These values can be modified again in interactive mode.

k Send a signal to an active process (aka kill a process).

q Quit the monitor program.

^F Show the next page of the process list (forward).

^B Show the previous page of the process list (backward).

RAW DATA STORAGE

In order to store system- and process level statistics for long-term analysis (e.g. to check the system load and the active processes running yesterday between 3:00 and 4:00 PM), *atop* can store the system- and process level statistics in compressed binary format in a raw file with the flag **-w** followed by the filename. If this file already exists and is recognized as a raw data file, *atop* will append new samples to the file (starting with a sample which reflects the activity since boot); if the file does not exist, it will be created.
By default only processes which have been active during the interval are stored in the raw file. When the flag **-a** is specified, all processes will be stored.
The interval (default: 10 seconds) and number of samples (default: infinite) can be passed as last arguments. Instead of the number of samples, the flag **-S** can be used to indicate that *atop* should finish just before midnight.

A raw file can be read and visualized again with the flag **-r** followed by the filename. If no filename is specified, the file **/var/opt/UNItools/log/atop"/atop_***YYYYMMDD* is opened for input (where *YYYYMMDD* are digits representing the current date). If a filename is specified in the format YYYYMMDD (representing any valid date), the file **/var/opt/UNItools/log/atop"/atop_***YYYYMMDD* is opened.
The samples from the file can be viewed interactively by using the key 't' to show the next sample and the key 'T' to show the previous sample. When output is redirected to a file or pipe, **atop** prints all samples in plain ASCII.
With the flag **-b** (begin time) and/or **-e** (end time) followed by a time argument of the form HH:MM, a certain time period within the raw file can be selected.

When **atop** is installed, two scripts are stored in the */etc/atop* directory. Each of these scripts take care that **atop** is activated every day to write compressed binary data to the file **/var/opt/UNItools/log/atop"/atop_***YYYYMMDD* with an interval of 10 minutes.
Furthermore the script removes all raw files which are older than four weeks.

Only one of these scripts should be used for automatic storage of the system- and process level information:

atop.daily This script should be used for systems on which process accounting is *not* activated via **logrotate** (i.e. the file **/etc/logrotate.d/psacct** is not present). In that case the script **atop.daily** can be activated every day (at midnight) via the **cron** daemon by creating the file */etc/cron.d/atop* with the contents

1 0 * * * root /etc/atop/atop.daily

atop.24hours For systems on which process accounting is daily restarted via **logrotate** the script **atop.24hours** should be used:

The section 'postrotate' in the file **/etc/logrotate.d/psacct** should be extended by calling the script **/etc/atop/atop.24hours** (without arguments) *after* reactivating process accounting with the **accton** command.

OUTPUT DESCRIPTION

The first sample shows the system level activity since boot (the elapsed time in the header shows the number of seconds since boot). Note that particular counters could have reached their maximum value (several times) and started by zero again, so do not rely on these figures.

For every sample *atop* first shows the lines related to system level activity. If a particular system resource has not been used during the interval, the entire line related to this resource is suppressed. So the number of system level lines may vary for each sample.
After that a list is shown of processes which have been active during the last interval. This list is by default sorted on cpu consumption, but this order can be changed by the keys which are previously described.

If values have to be shown by *atop* which do not fit in the column width, another notation is used. If e.g. a cpu-consumption of 233216 milliseconds should be shown in a column width of 4 positions, it is shown as '233s' (in seconds). For large memory figures, another unit is chosen if the value does not fit (Mb instead of Kb, Gb instead of Mb). For other values, a kind of exponent notation is used (value 123456789 shown in a column of 5 positions gives 123e6).

The system level information consists of the following output lines:

PRC

Process level totals.
This line contains the total cpu time consumed in system mode ('sys') and in user mode ('user'), the total number of processes present at this moment ('#proc'), the number of zombie processes ('#zombie') and the number of processes that ended during the interval ('#exit', which shows '?' if process accounting could not be switched on).

CPU

CPU utilization.
One line is shown for the total occupation of all CPU's together. In case of a multi-processor system, an additional line is shown for every individual processor (with 'cpu' in lower case), sorted on activity. Inactive cpu's will not be shown by default. The lines showing the per-cpu occupation contain the cpu number in the last field.
Every line contains the percentage of cpu time spent in kernel mode by all active processes ('sys'), the percentage of cpu time consumed in user mode ('user') for all active processes (including processes running with a nice value larger than zero), the percentage of cpu time spent for interrupt handling ('irq') including softirq, the percentage of unused cpu time while no processes were waiting for disk-I/O ('idle'), and the percentage of unused cpu time while at least one process was waiting for disk-I/O ('wait').
In case of per-cpu occupation, the last column shows the cpu number and the wait percentage ('w') for that cpu.

The number of lines showing the per-cpu occupation can be limited.

CPL CPU load information.
This line contains the load average figures reflecting the number of threads that are available to run on a CPU (i.e. part of the runqueue) or that are waiting for disk I/O. These figures are averaged over 1 ('avg1'), 5 ('avg5') and 15 ('avg15') minutes.
Furthermore the number of context switches ('csw') and the number of serviced interrupts ('intr') are shown.

MEM

Memory occupation.
This line contains the total amount of physical memory ('tot'), the amount of memory which is currently free ('free'), the amount of memory in use as page cache ('cache'), the amount of memory

used for filesystem meta data (‘buff’) and the amount of memory being used for kernel malloc’s (‘slab’ - always 0 for kernel 2.4).

SWP

Swap occupation and overcommit info.
This line contains the total amount of swap space on disk (‘tot’) and the amount of free swap space (‘free’).
Furthermore the committed virtual memory space (‘vmcom’) and the maximum limit of the committed space (‘vmlim’, which is by default swap size plus 50% of memory size) is shown. The committed space is the reserved virtual space for all allocations of private memory space for processes. The kernel only verifies whether the committed space exceeds the limit if strict overcommit handling is configured (vm.overcommit_memory is 2).

PAG Paging frequency.
This line contains the number of scanned pages (‘scan’) due to the fact that free memory drops below a particular threshold and the number times that the kernel tries to reclaim pages due to an urgent need (‘stall’).
Also the number of memory pages the system read from swap space (‘swin’) and the number of memory pages the system wrote to swap space (‘swout’) are shown.

DSK

Disk utilization.
Per active disk one line is produced, sorted on disk activity. Such line shows the name of the disk (e.g. hda or sda), the busy percentage i.e. the portion of time that the disk was busy handling requests (‘busy’), the number of read requests issued (‘read’), the number of write requests issued (‘write’) and the average number of milliseconds needed by a request (‘avio’) for seek, latency and data transfer.

The number of lines showing the disk occupation can be limited.

NET

Network utilization (TCP/IP).
One line is shown for activity of the transport layer (TCP and UDP), one line for the IP layer and one line per active interface.
For the transport layer, counters are shown concerning the number of received TCP segments including those received in error (‘tcpi’), the number of transmitted TCP segments excluding those containing only retransmitted octets (‘tcpo’), the number of UDP datagrams received (‘udpi’) and the number of UDP datagrams transmitted (‘udpo’). These counters are related to IPv4 and IPv6.
For the IP layer, counters are shown concerning the number of IP datagrams received from interfaces, including those received in error (‘ipi’), the number of IP datagrams that local higher-layer protocols offered for transmission (‘ipo’), the number of received IP datagrams which were forwarded to other interfaces (‘ipfrw’) and the number of IP datagrams which were delivered to local higher-layer protocols (‘deliv’). These counters are related to IPv4 and IPv6.
For every active network interface one line is shown, sorted on the interface activity. Such line shows the name of the interface and its busy percentage in the first column. The busy percentage for half duplex is determined by comparing the interface speed with the number of bits transmitted and received per second; for full duplex the interface speed is compared with the highest of either the transmitted or the received bits. When the interface speed can not be determined (e.g. for the loopback interface), ‘---’ is shown instead of the percentage.
Furthermore the number of received packets (‘pcki’), the number of transmitted packets (‘pcko’), the effective amount of bits received per second (‘si’) and the effective amount of bits transmitted per second (‘so’).

The number of lines showing the network interfaces can be limited.

Following the system level information, the processes are shown from which the resource utilization has

changed during the last interval. These processes might have used cpu time or issued disk- or network requests. However a process is also shown if part of it has been paged out due to lack of memory (while the process itself was in sleep state).

Per process the following fields may be shown (in alphabetical order), depending on the current output mode as described in the section INTERACTIVE COMMANDS:

CMD The name of the process. This name can be surrounded by "less/greater than" signs (‘<name>’) which means that the process has finished during the last interval.
Behind the abbreviation ‘CMD’ in the header line, the current page number and the total number of pages of the process list are shown.

COMMAND-LINE
The full command line of the process (including arguments), which is limited to the length of the screen line. Th command line can be surrounded by "less/greater than" signs (‘<line>’) which means that the process has finished during the last interval.
Behind the verb ‘COMMAND-LINE’ in the header line, the current page number and the total number of pages of the process list are shown.

CPU The occupation percentage of this process related to the available capacity for this resource on system level.

DSK The occupation percentage of this process related to the total load that is produced by all processes (i.e. total disk accesses by all processes during the last interval).
This information can only be shown when kernel patch ‘cnt’ is installed.

EXC The exit code of a terminated process (second position of column ‘ST’ is E) or the fatal signal number (second position of column ‘ST’ is S or C).

GROUP The real primary group identity under which the process runs.

MAJFLT
The number of page faults issued by this process.

MEM The occupation percentage of this process related to the available capacity for this resource on system level.

MINFLT The number of page reclaims issued by this process.

NET The occupation percentage of this process related to the total load that is produced by all processes (i.e. network packets transferred by all processes during the last interval).
This information can only be shown when kernel patch ‘cnt’ is installed.

NPROCS
The number of active and terminated processes accumulated for this user or program.

PID Process-id. If a process has been started and finished during the last interval, a ‘?’ is shown because the process-id is not part of the standard process accounting record. However when the kernel patch ‘acct’ is installed, this value will be shown.

POLICY Policy ’normal’ (SCHED_OTHER) refers to a timesharing process, ’fifo’ (SCHED_FIFO) and ’roundr’ (SCHED_RR) to a realtime process.

PRIO The process’ priority ranges from 0 (highest priority) to 139 (lowest priority). Priority 0 to 99 are used for realtime processes (fixed priority independent of their behavior) and priority 100 to 139 for timesharing processes (variable priority depending on their recent CPU consumption and the nice value).

RAWRS The number of raw datagrams received and sent by this process. This information can only be shown when kernel patch 'cnt' is installed.
If a process has finished during the last interval, no value is shown since network counters are not registered in the standard process accounting record. However when the kernel patch 'acct' is installed, this value will be shown.

RDDSK The number of read accesses issued physically on disk (so reading from the disk cache is not accounted for). This information can only be shown when kernel patch 'cnt' is installed.

RGROW The amount of resident memory that the process has grown during the last interval. A resident growth can be caused by touching memory pages which were not physically created/loaded before (load-on-demand). Note that a resident growth can also be negative e.g. when part of the process is paged out due to lack of memory or when the process frees dynamically allocated memory. For a process which started during the last interval, the resident growth reflects the total resident size of the process at that moment.
If a process has finished during the last interval, no value is shown since resident memory occupation is not part of the standard process accounting record. However when the kernel patch 'acct' is installed, this value will be shown.

RNET The number of TCP- and UDP packets received by this process. This information can only be shown when kernel patch 'cnt' is installed.
If a process has finished during the last interval, no value is shown since network counters are not part of the standard process accounting record. However when the kernel patch 'acct' is installed, this value will be shown.

RSIZE The total resident memory usage consumed by this process (or user).
If a process has finished during the last interval, no value is shown since resident memory occupation is not part of the standard process accounting record. However when the kernel patch 'acct' is installed, this value will be shown.

S The current state of the process: 'R' for running (currently processing or in the run queue), 'S' for sleeping interruptable (wait for an event to occur), 'D' for sleeping non-interruptable, 'Z' for zombie (waiting to be synchronized with its parent process), 'T' for stopped (suspended or traced), 'W' for swapping, and 'E' (exit) for processes which have finished during the last interval.

SNET The number of TCP- and UDP packets transmitted by this process. This information can only be shown when kernel patch 'cnt' is installed.
If a process has finished during the last interval, no value is shown since network-counters are not part of the standard process accounting record. However when the kernel patch 'acct' is installed, this value will be shown.

ST The status of a process.
The first position indicates if the process has been started during the last interval (the value *N* means 'new process').

The second position indicates if the process has been finished during the last interval.
The value *E* means 'exit' on the process' own initiative; the exit code is displayed in the column 'EXC'.
The value *S* means that the process has been terminated unvoluntarily by a signal; the signal number is displayed in the in the column 'EXC'.
The value *C* means that the process has been terminated unvoluntarily by a signal, producing a core dump in its current directory; the signal number is displayed in the in the column 'EXC'.

STDATE The start date of the process.

STTIME The start time of the process.

SYSCPU CPU time consumption of this process in system mode (kernel mode), usually due to system call handling.

TCPRCV

The number of receive requests issued by this process for TCP sockets, and the average size per transfer in bytes. This information can only be shown when kernel patch 'cnt' is installed.
If a process has finished during the last interval, no value is shown since network counters are not registered in the standard process accounting record. However when the kernel patch 'acct' is installed, this value will be shown.

TCPSND The number of send requests issued by this process for TCP sockets, and the average size per transfer in bytes. This information can only be shown when kernel patch 'cnt' is installed.
If a process has finished during the last interval, no value is shown since network counters are not registered in the standard process accounting record. However when the kernel patch 'acct' is installed, this value will be shown.

THR A multithreaded application consists of various threads. All related threads are contained in a thread group, represented by *atop* as one line.

On Linux 2.4 systems it is hardly possible to determine which threads (i.e. processes) are related to the same thread group. Every thread is represented by *atop* as a separate line.

UDPRCV

The number of UDP datagrams received by this process, and the average size per transfer in bytes. This information can only be shown when kernel patch 'cnt' is installed.
If a process has finished during the last interval, no value is shown since network counters are not registered in the standard process accounting record. However when the kernel patch 'acct' is installed, this value will be shown.

UDPSND

The number of UDP datagrams transmitted by this process, and the average size per transfer in bytes. This information can only be shown when kernel patch 'cnt' is installed.
If a process has finished during the last interval, no value is shown since network counters are not registered in the standard process accounting record. However when the kernel patch 'acct' is installed, this value will be shown.

USERNAME

The real user identity under which the process runs.

USRCPU

CPU time consumption of this process in user mode, due to processing the own program text.

VGROW The amount of virtual memory that the process has grown during the last interval. A virtual growth can be caused by e.g. issueing a malloc() or attaching a shared memory segment. Note that a virtual growth can also be negative by e.g. issueing a free() or detaching a shared memory segment. For a process which started during the last interval, the virtual growth reflects the total virtual size of the process at that moment.
If a process has finished during the last interval, no value is shown since virtual memory occupation is not part of the standard process accounting record. However when the kernel patch 'acct' is installed, this value will be shown.

VSIZE The total virtual memory usage consumed by this process (or user).
If a process has finished during the last interval, no value is shown since virtual memory occupation is not part of the standard process accounting record. However when the kernel patch 'acct'

is installed, this value will be shown.

VSTEXT
The virtual memory size used by the shared text of this process.

WRDSK The number of write accesses issued physically on disk (so writing to the disk cache is not accounted for). Usually application processes just transfer their data to the cache, while the physical write accesses are done later on by kernel daemons. This information can only be shown when kernel patch 'cnt' is installed.
Note that the number read- and write accesses are not separately maintained in the standard process accounting record. This means that only one value is given for read's and write's in case a process has finished during the last interval. However when the kernel patch 'acct' is installed, these values will be shown separately.

EXAMPLES

To monitor the current system load interactively with an interval of 5 seconds:

atop 5

To monitor the system load and write it to a file (in plain ASCII) with an interval of one minute during half an hour with active processes sorted on memory consumption:

atop -M 60 30 > /log/atop.mem

Store information about the system- and process activity in binary compressed form to a file with an interval of ten minutes during an hour:

atop -w /tmp/atop.raw 600 6

View the contents of this file:

atop -r /tmp/atop.raw

CONFIGURATION FILE

The default values used by **atop** can be overruled by a personal configuration file. This file, called **~/.atoprc** contains a keyword-value pair one every line (blank lines and lines starting with a #-sign are skipped). The following keywords can be specified:

flags A list of default flags can be defined here. The flags which are allowed are 'g', 'm', 'd', 'n', 'u', 'p', 's', 'c', 'v', 'C', 'M', 'D', 'N', 'A', 'a', 'f', '1' and 'x'.

interval The default interval value in seconds.

username
The default regular expression for the users for which active processes will be shown.

procname
The default regular expression for the process names to be shown.

maxlinecpu
The maximum number of active CPU's which will be shown.

maxlinedisk
The maximum number of active disks which will be shown.

maxlineintf
The maximum number of active network interfaces which will be shown.

cpucritperc
The busy percentage considered critical for a processor (see section COLORS). This percentage is used to determine a weighted percentage for line coloring and sorting of active processes. When this value is zero, no line coloring or automatic sorting is performed for this resource.

dskcritperc
The busy percentage considered critical for a disk (see section COLORS). This percentage is used to determine a weighted percentage for line coloring and sorting of active processes. When this value is zero, no line coloring or automatic sorting is performed for this resource.

netcritperc
The busy percentage considered critical for a network interface (see section COLORS). This percentage is used to determine a weighted percentage for line coloring and sorting of active processes. When this value is zero, no line coloring or automatic sorting is performed for this resource.

memcritperc
The percentage considered critical for memory utilization (see section COLORS). This percentage is used to determine a weighted percentage for line coloring and sorting of active processes. When this value is zero, no line coloring or automatic sorting is performed for this resource.

swpcritperc
The occupation percentage considered critical for swap space (see section COLORS). This percentage is used to determine a weighted percentage for line coloring and sorting of active processes. When this value is zero, no line coloring or automatic sorting is performed for this resource.

almostcrit
A percentage of the critical percentage to determine if the resource is almost critical (see section COLORS). When this value is zero, no line coloring for 'almost critical' is performed.

An example of the **~/.atoprc** file:

```
flags       af
interval    5
username
procname
maxlinecpu   4
maxlinedisk  10
maxlineintf  5
cpucritperc  80
almostcrit   90
```

FILES

/tmp/atop.d/atop.acct
File in which the kernel writes the accounting records if the standard accounting to the file **/var/log/pacct** or **/var/account/pacct** is not used.

~/.atoprc
Configuration file containing personal default values.

/var/opt/UNItools/log/atop"/atop_*YYYYMMDD*
Raw file, where *YYYYMMDD* are digits representing the current date. This name is used by the scripts **atop.daily** and **atop.24hours** as default name for the output file, and by **atop** as default name for the input file when using the **-r** flag.
All binary system- and process-level data in this file has been stored in compressed format.

SEE ALSO

atsar, logrotate
http://www.ATComputing.nl/atop, http://www.ATConsultancy.nl/atop

AUTHOR

Gerlof Langeveld, AT Computing (gerlof@ATComputing.nl)

NAME

bmon − Portable bandwidth monitor and rate estimator

SYNOPSIS

bmon [**−awShV**] [**−i** <*mod*>] [**−o** <*mod*>] [**−I** <*mod*>] [**−O** <*mod*>]
[**−f** <*path*>] [**−p** <*policy*>] [**−r** <*float*>] [**−s** <*float*>]

DESCRIPTION

bmon is a portable bandwidth monitor with multiple input methods and output modes. A set of architecture specific input modules provide the core with the listof interfaces and their counters. The core stores this counters and provides rate estimation including a history over the last 60 seconds, minutes, hours and days to the output modules which output them according to the configuration.

The set of counters is dependant on the input module and may vary. Secondary input and output modules may be used to collect counter values from other nodes or to write HTML statistics. This input/output architecture minimizes the work needed to port it to other architectures or generate specific statistics.

OPTIONS

−i Set primary *input module* and its configuration. The argument "list" will result in a list of available input modules. See INPUT MODULES for more details.

−o Set primary *output module* and its configuration. The argument "list" will result in a list of available output modules. See OUTPUT MODULES for more details.

−I Set secondary *input modules* and their configuration. The argument "list" will result in a list of available secondary input modules.

−O Set secondary *output modules* and their configuration. The argument "list" will result in a list of available secondary output modules.

−f Set alternative configuration *path*.

−p Set interface acceptance policy. See INTERFACE SELECTION for more details.

−a Include interfaces even if their status is down.

−r Set reading *interval* in which the input module will be called. The default for this is one second. Should be less or equal than 1 or a factor of it. Values not equal to 1 will result in additional rate calculation with the read interval as its unit.

−s Set sleeping *interval* between calls to output short interval callbacks for interactive output modules. Changing this can affect the variance of read intervals.

−w Enable signal driven output intervals. The output module will only be invoked upon receiving of SIGUSR1. Use bmon -S - to send the signal to a running bmon instance in signal driven mode.

−S Send SIGUSR1 to a running bmon instance. This arugment takes either - which will result in invoking ps to find bmon instances or a *pid* directly.

−h Prints a help text and exits.

−V Prints the version and exits.

INPUT MODULES

Input modules provide the core with interface statistics. Two kinds of modules exist, primary and secondary input modules. Their main difference is usage, there may be only one primary module running at the same time while the number of secondary input modules is not limited.

Every input module has a description, help text and list of options available which can be seen by adding the option "help" to the module options:

```
bmon -i netlink:help
```

See MODULE CONFIGURATION for mode details.

PRIMARY INPUT MODULES

netlink (Linux)

Requires libnl and uses an rtnetlink to collect interface statistics. This input module also provides statistics about traffic control qdiscs and classes. It is the preferred input module on Linux.

kstat (SunOS)

Provides interface statistics on SunOS operating systems in form of 32bit and 64bit counters. It is the preferred input module on SunOS.

sysctl (BSD, Darwin)

Provides interface statistics on BSD and Darwin operating systems. Is is the preferred input module on any BSD alike system.

proc (Linux)

Provides interface statistics on Linux using the proc filesystem (/proc/net/dev). It is one of the fallback input modules on Linux and will work on nearly every Linux kernel version.

sysfs (Linux)

Provides interface statistics on Linux using the sys filesystem (/sys/class/net/). It may be used together with newer Linux kernel versions but has no real advantage over the netlink input module. It caches open file descriptors to speed it up and is used as fallback method.

netstat (POSIX)

Provides limited interface statistics on almost any POSIX operating system by invoking netstat -i -a. Only use this as last hope.

dummy (any)

The purpose of the dummy input module is for testing. It generates in either a static or randomized form.

nulll (any)

Does not provide any interface statistics and thus can be used to disable local interface collection.

SECONDARY INPUT MODULES

distribution

Collects interface statistics from other nodes. It is the counterpart of the secondary output module called distribution. Its purpose is to distribute statistics in real time with not too much bandwidth consumption itself. See DISTRIBUTION for more details.

OUTPUT MODULES

Output modules are feeded with rate estimations and graphs from the core and print them out to the configured output device. Two kinds of modules exist, primary and secondary output modules. Their main difference is usage, there may be only one primary module running at the same time while the number of secondary output modules is not limited.

Every output module has a description, help text and list of options available which can be seen by adding the option "help" to the module options:

bmon -o ascii:help

See MODULE CONFIGURATION for mode details.

PRIMARY OUTPUT MODULES

ascii The ascii output modules prints out the diagrams and lists to standard output. The output format is highly configurable and suits as vmstat alike tool for interface statistics.

curses Interactive curses user interface providing real time rate estimations and graphs. The default view is a list of all interfaces grouped per node. The graphical history diagram and a list of detailed counters may be enabled/disable during runtime. Press '?' while the UI is running to see the quick reference.

SECONDARY OUTPUT MODULES

html Writes all interface statistics and diagrams as HTML files including a navigation menu for all nodes and their interfaces. The layout can be easly changes by altering the stylesheet which will not be overwritten.

distribution (any)
Distributes all statistics over the network using an UDP based statistic distribution protocol. The default configuration will use the multicast address all-nodes but it may also be configured so send to a unicast address.

MODULE CONFIGURATION

ARGUMENT ::= modulename:OPTS[,modulename:OPTS[,...]]
OPTS ::= OPTION[;OPTION[;...]]
OPTION ::= type[=value]

If you specify multiple primrary input or output modules the first reported to be working module will be used.

If you specify multiple secondary input or output modules all of them will get invoked.

DISTRIBUTION

Statistic distribution is a powerful method to spread the statistic all over the network and make the available on every machine. The advantage over web based statistic overviews and multi terminal remote shell based solutions is its nearly realtime accuracy while being lightweight and not polluting the network too much. The protocol is UDP based and thus not reliable and optmized on size.

See include/bmon/distribution.h for the protocol specification.

DIAGRAM TYPES

You will find the following diagram types being used by all output modules in some form:

list A list of interfaces including their byte and packets rate (bps/pps).

graphical history diagram
A graph showing the history of a counter over the last 60 (read interval/ seconds/minutes/hours/days). The outer left column is the most recent rate while the outer right column is the most outdated. The preferred diagram to impress co-workers.

detailed
Detailed counters such as error counters or other attributes assigned to this interface. The list of attributes may very depending on the input module and architecture of the host OS.

INTERFACE SELECTION

SELECTION ::= NAME[,NAME[,...]]
NAME ::= [!]interface

The interface name may contain the character '*' which will act as a wildcard and represents any number of any character type, i.e. eth*, h*0, ...

Examples:
lo,eth0,eth1
eth*,!eth0

CONFIGURATION FILE

Bmon will try and read configuration data from the following files in the specified order: /etc/bmon.conf, $HOME/.bmonrc.

None of the above files will be read if the path to the configuration file was specified using the -f option.

Configuration possibilities:

input <*module configuration*>
Specify primary input module (-i), see INPUT MODULES.

secondary_input <*module configuration*>
Specify secondary input modules (-I), see INPUT MODULES.

output <*module configuration*>
Specify primary output module (-o), see OUTPUT MODULES.

secondary_output <*module configuration*>
Specify secondary output modules (-O), see OUTPUT MODULES.

policy <*policy*>
Set interface acceptance policy (-p), see INTERFACE SELECTION.

read_interval <*interval*>
Set reading interval in which the input module will be called (-r).

sleep_time <*interval*>
Set sleeping interval between calls to output short interval callbacks for interactive output modules. (-s)

show_all
Include interface even if their status is down. (-a)

include <*file*>
Include *file* and read it as configuration file.

Color layouts
See COLOR LAYOUTS.

Bindings
See BIND INTERFACE.

COLOR LAYOUTS

The layout is used to specify the look'n'feel of the curses output module. The color "default" represents the terminal color which can be used to keep the background transparent for transparent terminals.

Colors: default, black, red, green, yellow, blue, magenta, cyan, white

Flags: reverse

Layouts: Default, Statusbar, Header, List, Selected,

Prototype:
Layout <*name*> <*foreground*> <*background*> <*flags*>

Example:
Layout Statusbar red black reverse

Feel free to submit patches extending the configurability using layouts.

BIND INTERFACE

The bind interface can be used to bind not yet assigned keys to shell scripts. It currently works in the curses output module but it might be ported to other output modules in the future. The interface name of the currently selected interface is provided to the script via the first argument.

Prototype:
Bind <*key*> <*Executable*>

Example:
bind D /sbin/intf_down.sh

EXAMPLES

To run bmon in curses mode monitoring the interfaces eth0 and eth1:
bmon -i eth0,eth1 -o curses

To run bmon in acii mode printing the detailed diagram with fixed y-axis unit:
bmon -o 'ascii:diagram=detailed;ynit=kb'

To run bmon in signal driven mode drawing the graphical diagram with customized drawing characters and fixed x and y axis:
bmon -s -o 'ascii:diagram=graph;fgchar=#;bgcar=_;xunit=min'

To run bmon with no primrary output (daemon) but distribute the statistic over the network:
bmon -o null -O distribution

To run bmon collecting local and remote statistics and show it in curses mode:

bmon -I distribution:multicast -o curses

To build a relay and collect remote statistic and send them to a unicast address while ignoring errors because the destination is not available:

bmon -i null -I distribution:multicast -o null -O 'distribution:ip=10.0.0.1;errignore;forward'

To collect local statistics and those from the whole network and generate a HTML page out of the those statistics:

bmon -I distribution:multicast -o null -O html:path=/var/istats/

KNOWN ISSUES

The curses output modules doesn't work properly on NetBSD < 2.0 because getch() cannot be set to be non-blocking.

sysctl input segfaults on sparc64 OpenBSD.

FILES

/etc/bmon.conf
$HOME/.bmonrc

SEE ALSO

ifconfig(8), kstat(1M), netlink(3)

AUTHOR

Thomas Graf <tgraf@suug.ch>

NAME

rx, rb, rz − XMODEM, YMODEM, ZMODEM (Batch) file receive

SYNOPSIS

rz [− **+8abeOpqRtTuUvy**]
rb [− **+abqRtuUvy**]
rx [− **abceqRtuUv**] *file*
[−][**v**]**rzCOMMAND**

DESCRIPTION

This program uses error correcting protocols to receive files over a dial-in serial port from a variety of programs running under PC-DOS, CP/M, Unix, and other operating systems. It is invoked from a shell prompt manually, or automatically as a result of an "sz file ..." command given to the calling program.

While *rz* is smart enough to be called from *cu(1),* very few versions of *cu(1)* are smart enough to allow *rz* to work properly. Unix flavors of Professional-YAM are available for such dial-out application.

Rz (Receive ZMODEM) receives files with the ZMODEM batch protocol. Pathnames are supplied by the sending program, and directories are made if necessary (and possible). Normally, the "rz" command is automatically issued by the calling ZMODEM program, but some defective ZMODEM implementations may require starting *rz* the old fashioned way.

Rb receives file(s) with YMODEM, accepting either standard 128 byte sectors or 1024 byte sectors (YAM sb **-k** option). The user should determine when the 1024 byte block length actually improves throughput without causing lost data or even system crashes.

If True YMODEM (Omen Technology trademark) file information (file length, etc.) is received, the file length controls the number of bytes written to the output dataset, and the modify time and file mode (iff non zero) are set accordingly.

If no True YMODEM file information is received, slashes in the pathname are changed to underscore, and any trailing period in the pathname is eliminated. This conversion is useful for files received from CP/M systems. With YMODEM, each file name is converted to lower case unless it contains one or more lower case letters.

Rx receives a single *file* with XMODEM or XMODEM-1k protocol. The user should determine when the 1024 byte block length actually improves throughput without causing problems. The user must supply the file name to both sending and receiving programs. Up to 1023 garbage characters may be added to the received file.

Rz may be invoked as **rzCOMMAND** (with an optional leading − as generated by login(1)). For each received file, *rz* will pipe the file to ‘‘COMMAND filename’’ where filename is the name of the transmitted file with the file contents as standard input.

Each file transfer is acknowledged when COMMAND exits with 0 status. A non zero exit status terminates transfers.

A typical use for this form is *rzrmail* which calls rmail(1) to post mail to the user specified by the transmitted file name. For example, sending the file "caf" from a PC-DOS system to *rzrmail* on a Unix system would result in the contents of the DOS file "caf" being mailed to user "caf".

On some Unix systems, the login directory must contain a link to COMMAND as login sets SHELL=rsh

which disallows absolute pathnames. If invoked with a leading ''v'', *rz* will be verbose (see **v** option). The following entry works for Unix SYS III/V:

rzrmail::5:1::/bin:/usr/local/rzrmail

If the SHELL environment variable includes *rsh* , *rbash* or *rksh* (restricted shell), *rz* will not accept absolute pathnames or references to a parent directory, will not modify an existing file, and removes any files received in error.

If **rz** is invoked with stdout and stderr to different datasets, Verbose is set to 2, causing frame by frame progress reports to stderr. This may be disabled with the **q** option.

OPTIONS

The meanings of the available options are:

-+, --append
append received data to an existing file (ZMODEM, ASCII only).

-a, --ascii
Convert files to Unix conventions by stripping carriage returns and all characters beginning with the first Control Z (CP/M end of file).

-b, --binary
Binary (tell it like it is) file transfer override.

-B NUMBER, --bufsize NUMBER
Buffer **NUMBER** bytes before writing to disk. Default ist 32768, which should be enough for most situations. If you have a slow machine or a bad disk interface or suffer from other hardware problems you might want to increase the buffersize. **-1** or **auto** use a buffer large enough to buffer the whole file. Be careful with this options - things normally get worse, not better, if the machine starts to swap.

-c, --with-crc
XMODEM only. Use 16 bit CRC (normally a one byte checksum is used).

-C, --allow-remote-commands
allow remote command execution (**insecure**). This allows the sender to execute an arbitrary command through **system** () or **execl** (). Default is to disable this feature (?). This option is ignored if running in restricted mode.

-D, --null
Output file data to /dev/null; for testing. (Unix only)

--delay-startup N
Wait **N** seconds before doing anything.

-e, --escape
Force sender to escape all control characters; normally XON, XOFF, DLE, CR-@-CR, and Ctrl-X are escaped.

-E, --rename
Rename incoming file if target filename already exists. The new file name will have a dot and a number (0..999) appended.

-h, --help
give help screen.

-m N, --min-bps N
Stop transmission if BPS-Rate (Bytes Per Second) falls below N for a certain time (see --min-bps-time option).

-M N, --min-bps-time
Used together with --min-bps. Default is 120 (seconds).

-O, --disable-timeouts
Disable read timeout handling code. This makes lrz hang if the sender does not send any more, but increases performance (a bit) and decreases system load (through reducing the number of system calls by about 50 percent).

Use this option with care.

--o-sync

Open output files in synchronous write mode. This may be useful if you experience errors due to lost interrupts if update (or bdflush or whoever this daemon is called on your system) writes the buffers to the disk.

This option is ignored and a warning is printed if your systems doesn't support O_SYNC.

-p, --protect

(ZMODEM) Protect: skip file if destination file exists.

-q, --quiet

Quiet suppresses verbosity.

-r, --resume

Crash recovery mode. lrz tries to resume interrupted file transfers.

-R, --restricted

Enter more restricted mode. lrz will not create directories or files with a leading dot if this option is given twice.

See **SECURITY** for mode information about restricted mode.

-s HH:MM, --stop-at HH:MM

Stop transmission at **HH** hours, **MM** minutes. Another variant, using **+N** instead of **HH:MM,** stops transmission in **N** seconds.

-S, --timesync

Request timesync packet from the sender. The sender sends its system time, causing lrz to complain about more then 60 seconds difference.

Lrz tries to set the local system time to the remote time if this option is given twice (this fails if lrz is not run by root).

This option makes lrz incompatible with certain other ZModems. Don't use it unless you know what you are doing.

--syslog[=off]

turn syslogging on or off. the default is set at configure time. This option is ignored if no syslog support is compiled in.

-t TIM, --timeout TIM

Change timeout to *TIM* tenths of seconds. This is ignored if timeout handling is turned of through the **O** option.

--tcp-client ADDRESS:PORT

Act as a tcp/ip client: Connect to the given port.

See **--tcp-server** for more information.

--tcp-server

Act as a server: Open a socket, print out what to do, wait for connection.

You will normally not want to use this option as lrzsz is the only zmodem which understands what to do (private extension). You might want to use this if you have to use zmodem (for which reason whatever), and cannot use the **--tcp** option of *lsz* (perhaps because your telnet doesn't allow to spawn a local program with stdin/stdout connected to the remote side).

If you use this option you have to start *lsz* with the **--tcp-client ADDRESS:PORT** option. *lrz will print the address and port on startup.*

Use of this option imposes a security risk, somebody else could connect to the port in between. See **SECURITY** for details.

-U, --unrestrict
turn off restricted mode (this is not possible if running under a restricted shell).
--version
prints out version number.
-v, --verbose
Verbose causes a list of file names to be appended to stderr. More v's generate more output.
-wN, --windowsize N
Set window size to N.
-X, --xmodem
use XMODEM protocol.
-y, --overwrite
Yes, clobber any existing files with the same name.
--ymodem
use YMODEM protocol.
-Z, --zmodem
use ZMODEM protocol.

SECURITY

Contrary to the original ZMODEM lrz defaults to restricted mode. In restricted mode lrz will not accept absolute pathnames or references to a parent directory, will not modify an existing file, and removes any files received in error. Remote command execution is disabled.

To use a more restricted mode set the environment variable **ZMODEM_RESTRICTED** or give the **R** option. This disables creation of subdirectories and invisible files.

Restricted mode may be turned off with the **U** option, unless lrz runs under a restricted shell.

Use of the
--tcp-client or **--tcp-server** options imposes a security risk, as somebody else could connect to the port before you do it, and grab your data. If there's strong demand for a more secure mode i might introduce some sort of password challenge.

ENVIRONMENT

lrz uses the following environment variables:

SHELL
lrz recognizes a restricted shell if this variable includes *rsh* or *rksh*

ZMODEM_RESTRICTED
lrz enters the more restricted mode if the variable is set.

EXAMPLES

(Pro-YAM command)
<ALT-2>
Pro-YAM Command: *sz *.h *.c*
(This automatically invokes *rz* on the connected system.)

SEE ALSO

ZMODEM.DOC, YMODEM.DOC, Professional-YAM, crc(omen), sz(omen), usq(omen), undos(omen)

Compile time options required for various operating systems are described in the source file.

NOTES

Sending serial data to timesharing minicomputers at sustained high speeds has been known to cause lock-ups, system halts, kernel panics, and occasional antisocial behaviour. When experimenting with high speed input to a system, consider rebooting the system if the file transfers are not successful, especially if the personality of the system appears altered.

The Unix "ulimit" parameter must be set high enough to permit large file transfers.

The TTY input buffering on some systems may not allow long blocks or streaming input at high speed. You should suspect this problem when you can't send data to the Unix system at high speeds using ZMODEM, YMODEM-1k or XMODEM-1k, when YMODEM with 128 byte blocks works properly. If the system's tty line handling is really broken, the serial port or the entire system may not survive the onslaught of long bursts of high speed data.

The DSZ or Pro-YAM **zmodem l** numeric parameter may be set to a value between 64 and 1024 to limit the burst length ("zmodem pl128").

32 bit CRC code courtesy Gary S. Brown. Directory creation code from John Gilmore's PD TAR program.

BUGS

Calling *rz* from most versions of cu(1) doesn't work because cu's receive process fights *rz* for characters from the modem.

Programs that do not properly implement the specified file transfer protocol may cause *sz* to "hang" the port for a minute or two. Every reported instance of this problem has been corrected by using ZCOMM, Pro-YAM, or other program with a correct implementation of the specified protocol.

Many programs claiming to support YMODEM only support XMODEM with 1k blocks, and they often don't get that quite right.

Pathnames are restricted to 127 characters. In XMODEM single file mode, the pathname given on the command line is still processed as described above. The ASCII option´s CR/LF to NL translation merely deletes CR´s; undos(omen) performs a more intelligent translation.

VMS VERSION

The VMS version does not set the file time.

VMS C Standard I/O and RMS may interact to modify file contents unexpectedly.

The VMS version does not support invocation as **rzCOMMAND .** The current VMS version does not support XMODEM, XMODEM-1k, or YMODEM.

According to the VMS documentation, the buffered input routine used on the VMS version of *rz* introduces a delay of up to one second for each protocol transaction. This delay may be significant for very short files. Removing the "#define BUFREAD" line from rz.c will eliminate this delay at the expense of increased CPU utilization.

The VMS version causes DCL to generate a random off the wall error message under some error conditions; this is a result of the incompatibility of the VMS "exit" function with the Unix/MSDOS standard.

ZMODEM CAPABILITIES

Rz supports incoming ZMODEM binary (-b), ASCII (-a), protect (-p), clobber (-y), and append (-+) requests. The default is protect (-p) and binary (-b).

The Unix versions support ZMODEM command execution.

FILES

rz.c, crctab.c, rbsb.c, zm.c, zmodem.h Unix source files.

rz.c, crctab.c, vrzsz.c, zm.c, zmodem.h, vmodem.h, vvmodem.c, VMS source files.

NAME

sx, sb, sz − XMODEM, YMODEM, ZMODEM file send

SYNOPSIS

sz [−**+8abdefkLlNnopqTtuvyY**] *file ...*
sb [−**adfkqtuv**] *file ...*
sx [−**akqtuv**] *file*
sz [−**oqtv**] **-c COMMAND**
sz [−**oqtv**] **-i COMMAND**
sz -TT

DESCRIPTION

Sz uses the ZMODEM, YMODEM or XMODEM error correcting protocol to send one or more files over a dial-in serial port to a variety of programs running under PC-DOS, CP/M, Unix, VMS, and other operating systems.

While *rz* is smart enough to be called from *cu(1),* very few versions of *cu(1)* are smart enough to allow *sz* to work properly. Unix flavors of Professional-YAM are available for such dial-out application.

Sz sends one or more files with ZMODEM protocol.

ZMODEM greatly simplifies file transfers compared to XMODEM. In addition to a friendly user interface, ZMODEM provides Personal Computer and other users an efficient, accurate, and robust file transfer method.

ZMODEM provides complete **END-TO-END** data integrity between application programs. ZMODEM's 32 bit CRC catches errors that sneak into even the most advanced networks.

Advanced file management features include AutoDownload (Automatic file Download initiated without user intervention), Display of individual and total file lengths and transmission time estimates, Crash Recovery, selective file transfers, and preservation of exact file date and length.

Output from another program may be piped to **sz** for transmission by denoting standard input with "-":

ls -l | sz -

The program output is transmitted with the filename sPID.sz where PID is the process ID of the **sz** program. If the environment variable **ONAME** is set, that is used instead. In this case, the Unix command:

ls -l | ONAME=con sz -ay -

will send a "file" to the PC-DOS console display. The **-y** option instructs the receiver to open the file for writing unconditionally. The **-a** option causes the receiver to convert Unix newlines to PC-DOS carriage returns and linefeeds.

Sb batch sends one or more files with YMODEM or ZMODEM protocol. The initial ZMODEM initialization is not sent. When requested by the receiver, **sb** supports **YMODEM-g** with "cbreak" tty mode, XON/XOFF flow control, and interrupt character set to CAN (^X). **YMODEM-g** (Professional-YAM **g** option) increases throughput over error free channels (direct connection, X.PC, etc.) by not acknowledging each transmitted sector.

On Unix systems, additional information about the file is transmitted. If the receiving program uses this information, the transmitted file length controls the exact number of bytes written to the output dataset, and the modify time and file mode are set accordingly.

Sx sends a single *file* with **XMODEM** or **XMODEM-1k** protocol (sometimes incorrectly called "ymodem"). The user must supply the file name to both sending and receiving programs.

If **sz** is invoked with $SHELL set and iff that variable contains the string *rsh* , *rbash* or *rksh* (restricted shell), **sz** operates in restricted mode. Restricted mode restricts pathnames to the current directory and PUBDIR (usually /usr/spool/uucppublic) and/or subdirectories thereof.

The fourth form sends a single COMMAND to a ZMODEM receiver for execution. **Sz** exits with the COMMAND return value. If COMMAND includes spaces or characters special to the shell, it must be quoted.

The fifth form sends a single COMMAND to a ZMODEM receiver for execution. **Sz** exits as soon as the receiver has correctly received the command, before it is executed.

The sixth form (sz -TT) attempts to output all 256 code combinations to the terminal. In you are having difficulty sending files, this command lets you see which character codes are being eaten by the operating system.

If **sz** is invoked with stdout and stderr to different datasets, Verbose is set to 2, causing frame by frame progress reports to stderr. This may be disabled with the **q** option.

The meanings of the available options are:

-+, --append
: Instruct the receiver to append transmitted data to an existing file (ZMODEM only).

-2, --twostop
: use two stop bits (if possible). Do not use this unless you know what you are doing.

-8, --try-8k
: Try to go up to 8KB blocksize. This is incompatible with standard zmodem, but a common extension in the bbs world. (ZMODEM only).

--start-8k
: Start with 8KB blocksize. Like --try-8k.

-a, --ascii
: Convert NL characters in the transmitted file to CR/LF. This is done by the sender for XMODEM and YMODEM, by the receiver for ZMODEM.

-b, --binary
: (ZMODEM) Binary override: transfer file without any translation.

-B NUMBER, --bufsize NUMBER
: Use a readbuffer of **NUMBER** bytes. Default ist 16384, which should be enough for most situations. If you have a slow machine or a bad disk interface or suffer from other hardware problems you might want to increase the buffersize. **-1** or **auto** use a buffer large enough to buffer the whole file. Be careful with this option - things normally get worse, not better, if the machine starts to swap.

  Using this option turns of memory mapping of the input file. This increases memory and cpu usage.

-c COMMAND, --command COMMAND
: Send COMMAND to the receiver for execution, return with COMMAND´s exit status.

-C N, --command-tries N
: Retry to send command N times (default: 11).

-d, --dot-to-slash
: Change all instances of "." to "/" in the transmitted pathname. Thus, C.omenB0000 (which is unacceptable to MSDOS or CP/M) is transmitted as C/omenB0000. If the resultant filename has more than 8 characters in the stem, a "." is inserted to allow a total of eleven.

This option enables the **--full-path** option.

--delay-startup N

Wait **N** seconds before doing anything.

-e, --escape

Escape all control characters; normally XON, XOFF, DLE, CR-@-CR, and Ctrl-X are escaped.
Force the sender to rename the new file if a file with the same
name already exists.

-f, --full-path

Send Full pathname. Normally directory prefixes are stripped from the transmitted filename.

This is also turned on with to **--dot-to-slash** option.

-h, --help

give help.

-i COMMAND, --immediate-command COMMAND

Send COMMAND to the receiver for execution, return immediately upon the receiving program's successful recption of the command.

-k, --1k

(XMODEM/YMODEM) Send files using 1024 byte blocks rather than the default 128 byte blocks. 1024 byte packets speed file transfers at high bit rates. (ZMODEM streams the data for the best possible throughput.)

-L N, --packetlen N

Use ZMODEM sub-packets of length N. A larger N (32 <= N <= 1024) gives slightly higher throughput, a smaller N speeds error recovery. The default is 128 below 300 baud, 256 above 300 baud, or 1024 above 2400 baud.

-m N, --min-bps N

Stop transmission if BPS-Rate (Bytes Per Second) falls below N for a certain time (see --min-bps-time option).

-M N, --min-bps-time

Used together with --min-bps. Default is 120 (seconds).

-l N, --framelen N

Wait for the receiver to acknowledge correct data every **N** (32 <= N <= 1024) characters. This may be used to avoid network overrun when XOFF flow control is lacking.

-n, --newer

(ZMODEM) Send each file if destination file does not exist. Overwrite destination file if source file is newer than the destination file.

-N, --newer-or-longer

(ZMODEM) Send each file if destination file does not exist. Overwrite destination file if source file is newer or longer than the destination file.

-o, --16-bit-crc

(ZMODEM) Disable automatic selection of 32 bit CRC.

-O, --disable-timeouts

Disable read timeout handling. This makes lsz hang if the other side doesn't send anything, but increases performance (not much) and decreases system load (reduces number of system calls by about 50 percent).

Use this option with care.

-p, --protect

(ZMODEM) Protect existing destination files by skipping transfer if the destination file exists.

-q, --quiet

Quiet suppresses verbosity.

-R, --restricted

Restricted mode: restricts pathnames to the current directory and PUBDIR (usually /usr/spool/uucppublic) and/or subdirectories thereof.

-r, --resume
(ZMODEM) Resume interrupted file transfer. If the source file is longer than the destination file, the transfer commences at the offset in the source file that equals the length of the destination file.

-s HH:MM, --stop-at HH:MM
Stop transmission at **HH** hours, **MM** minutes. Another variant, using **+N** instead of **HH:MM,** stops transmission in **N** seconds.

-S, --timesync
enable timesync protocol support. See timesync.doc for further information.

This option is incompatible with standard zmodem. Use it with care.

--syslog[=off]
turn syslogging on or off. the default is set at configure time. This option is ignored if no syslog support is compiled in.

-t TIM, --timeout TIM
Change timeout to *TIM* tenths of seconds.

-T, --turbo
Do not escape certain characters (^P, ^P|0x80, telenet escape sequence [CR + @]). This improves performance by about 1 percent and shouldn't hurt in the normal case (but be careful - ^P might be useful if connected through a terminal server).

--tcp Try to initiate a TCP/IP connection. lsz will ask the receiving zmodem to open a TCP/IP connection. All handshaking (which address / port to use) will be done by the zmodem programs.

You will normally not want to use this option as lrzsz is the only zmodem which understands what to do (private extension). You might want to use this option if the two programs are connected (stdin/out) over a slow or bad (not 8bit clean) network connection.

Use of this option imposes a security risk, somebody else could connect to the port in between. See **SECURITY** for details.

--tcp-client ADDRESS:PORT
Act as a tcp/ip client: Connect to the given port.

See **--tcp-server** for more information.

--tcp-server
Act as a server: Open a socket, print out what to do, wait for connection.

You will normally not want to use this option as lrzsz is the only zmodem which understands what to do (private extension). You might want to use this if you have to use zmodem (for which reason whatever), and cannot use the **--tcp** option of *lsz* (perhaps because your telnet doesn't allow to spawn a local program with stdin/stdout connected to the remote side).

If you use this option you have to start *lsz* with the **--tcp-client ADDRESS:PORT** option. *lrz will print the address and port on startup.*

Use of this option imposes a security risk, somebody else could connect to the port in between. See **SECURITY** for details.

-u Unlink the file after successful transmission.

-U, --unrestrict
Turn off restricted mode (this is not possible if running under a restricted shell).

-w N, --windowsize N
Limit the transmit window size to N bytes (ZMODEM).

-v, --verbose
Verbose output to stderr. More v's generate more output.

-X, --xmodem
use XMODEM protocol.

-y, --overwrite
Instruct a ZMODEM receiving program to overwrite any existing file with the same name.

-Y, --overwrite-or-skip
Instruct a ZMODEM receiving program to overwrite any existing file with the same name, and to skip any source files that do have a file with the same pathname on the destination system.

--ymodem
use ZMODEM protocol.

-Z, --zmodem
use ZMODEM protocol.

SECURITY

Restricted mode restricts pathnames to the current directory and PUBDIR (usually /var/spool/uucppublic) and/or subdirectories thereof, and disables remote command execution.

Restricted mode is entered if the **R** option is given or if lsz detects that it runs under a restricted shell or if the environment variable ZMODEM_RESTRICTED is found.

Restricted mode can be turned of with the **U** option if not running under a restricted shell.

Use of the
--tcp-client or **--tcp-server** options imposes a security risk, as somebody else could connect to the port before you do it, and grab your data. If there's strong demand for a more secure mode i might introduce some sort of password challenge.

ENVIRONMENT

ZNULLS
may be used to specify the number of nulls to send before a ZDATA frame.

SHELL
lsz recognizes a restricted shell if this variable includes *rsh* or *rksh*

ZMODEM_RESTRICTED
lrz enters restricted mode if the variable is set.

TMPDIR
If this environment variable is set its content is used as the directory to place in the answer file to a **timesync** request. **TMP** Used instead of TMPDIR if TMPDIR is not set. If neither TMPDIR nor TMP is set /tmp will be used.

EXAMPLES

ZMODEM File Transfer (Unix to DSZ/ZCOMM/Professional-YAM)
% sz −a *.c
This single command transfers all .c files in the current Unix directory with conversion (**−a**) to end of line conventions appropriate to the receiving environment. With ZMODEM AutoDownload enabled, Professional-YAM and ZCOMM will automatically recieve the files after performing a security check.

% sz −Yan *.c *.h
Send only the .c and .h files that exist on both systems, and are newer on the sending system than the corresponding version on the receiving system, converting Unix to DOS text format.
$ sz -\Yan file1.c file2.c file3.c foo.h baz.h ®(for VMS)

ZMODEM Command Download (Unix to Professional-YAM)

```
cpszall:all
  sz −c "c:;cd /yam/dist"
  sz −ya $(YD)/*.me
  sz −yqb y*.exe
  sz −c "cd /yam"
  sz −i "!insms"
```

This Makefile fragment uses **sz** to issue commands to Professional-YAM to change current disk and directory. Next, **sz** transfers the *.me* files from the $YD directory, commanding the receiver to overwrite the old files and to convert from Unix end of line conventions to PC-DOS conventions. The third line transfers some *.exe* files. The fourth and fifth lines command Pro-YAM to change directory and execute a PC-DOS batch file *insms* . Since the batch file takes considerable time, the **−i** form is used to allow **sz** to exit immediately.

XMODEM File Transfer (Unix to Crosstalk)

% **sx −a foo.c**
ESC
rx foo.c

The above three commands transfer a single file from Unix to a PC and Crosstalk with *sz* translating Unix newlines to DOS CR/LF. This combination is much slower and far less reliable than ZMODEM.

ERROR MESSAGES

"Caught signal 99" indicates the program was not properly compiled, refer to "bibi(99)" in rbsb.c for details.

SEE ALSO

rz(omen), ZMODEM.DOC, YMODEM.DOC, Professional-YAM, crc(omen), sq(omen), todos(omen), tocpm(omen), tomac(omen), yam(omen)

Compile time options required for various operating systems are described in the source file.

VMS VERSION

The VMS version does not support wild cards. Because of VMS DCL, upper case option letters muse be represented by \ proceding the letter.

The current VMS version does not support XMODEM, XMODEM-1k, or YMODEM.

VMS C Standard I/O and RMS may interact to modify the file contents.

FILES

32 bit CRC code courtesy Gary S. Brown.

sz.c, crctab.c, rbsb.c, zm.c, zmodem.h Unix source files

sz.c, crctab.c, vrzsz.c, zm.c, zmodem.h, vmodem.h, vvmodem.c, VMS source files.

/tmp/szlog stores debugging output (sz -vv) (szlog on VMS).

TESTING FEATURE

The command "sz -T file" exercises the **Attn** sequence error recovery by commanding errors with unterminated packets. The receiving program should complain five times about binary data packets being too long. Each time **sz** is interrupted, it should send a ZDATA header followed by another defective packet. If the receiver does not detect five long data packets, the **Attn** sequence is not interrupting the sender, and the **Myattn** string in **sz.c** must be modified.

After 5 packets, **sz** stops the "transfer" and prints the total number of characters "sent" (Tcount). The difference between Tcount and 5120 represents the number of characters stored in various buffers when the Attn sequence is generated.

BUGS

Calling *sz* from most versions of cu(1) doesn't work because cu's receive process fights *sz* for characters from the modem.

On at least one BSD system, sz would hang or exit when it got within a few kilobytes of the end of file. Using the "-w 8192" flag fixed the problem. The real cause is unknown, perhaps a bug in the kernel TTY output routines.

Programs that do not properly implement the specified file transfer protocol may cause *sz* to "hang" the port for a minute or two. This problem is corrected by using ZCOMM, Pro-YAM, or other program with a correct implementation of the specified protocol.

Many programs claiming to support YMODEM only support XMODEM with 1k blocks, and they often don't get that quite right.

XMODEM transfers add up to 127 garbage bytes per file. XMODEM-1k and YMODEM-1k transfers use 128 byte blocks to avoid extra padding.

YMODEM programs use the file length transmitted at the beginning of the transfer to prune the file to the correct length; this may cause problems with source files that grow during the course of the transfer. This problem does not pertain to ZMODEM transfers, which preserve the exact file length unconditionally.

Most ZMODEM options are merely passed to the receiving program; some do not implement all these options.

Circular buffering and a ZMODEM sliding window should be used when input is from pipes instead of acknowledging frames each 1024 bytes. If no files can be opened, **sz** sends a ZMODEM command to echo a suitable complaint; perhaps it should check for the presence of at least one accessible file before getting hot and bothered. The test mode leaves a zero length file on the receiving system.

A few high speed modems have a firmware bug that drops characters when the direction of high speed transmissson is reversed. The environment variable ZNULLS may be used to specify the number of nulls to send before a ZDATA frame. Values of 101 for a 4.77 mHz PC and 124 for an AT are typical.

NAME

netserver − a network performance benchmark server

SYNOPSIS

netserver [-P portnum] [-n numcpus]

DESCRIPTION

Netserver listens for connections from a benchmark, and responds accordingly. It can either be run from or as a standalone daemon (with the -p flag). If run from the -p option should not be used.

OPTIONS

−h Display a usage string, and exit.

−n numcpus

Specify the number of CPU's in the system on those systems for which netperf has no way to find the number of CPU's programatically.

−p portnum

Listen on the specified port. This is used when running as a standalone daemon.

BUGS

No known bugs at this time. If you think you have found a bug, please send email to Rick Jones <raj@cup.hp.com>.

SEE ALSO

Netperf: A Network Performance Benchmark
http://www.netperf.org/

AUTHORS

HP Information Networks Division - Networking Performance Team.
Rick Jones <raj@cup.hp.com>
Karen Choy HP IND
Dave Shield <daves@csc.liv.ac.uk> (man pages)
Others too numerous to mention here - see the ACKNWLDGMNTS file

NAME

netperf − a network performance benchmark

SYNOPSIS

netperf [global options] -- [test specific options]

DESCRIPTION

Netperf is a benchmark that can be used to measure various aspects of networking performance. Currently, its focus is on bulk data transfer and request/response performance using either TCP or UDP, and the Berkeley Sockets interface. In addition, tests for DLPI, and Unix Domain Sockets, tests for IPv6 may be conditionally compiled-in.

GLOBAL OPTIONS

−a sizespec

Alter the send and receive buffer alignments on the local system. This defaults to 8 bytes.

−A sizespec

As -a, but for the remote system.

−c [rate]

Request CPU utilization and service demand calculations for the local system. If the optional rate parameter is specified, **netperf** will use that instead of calculating the rate itself.

−C [rate]

As -c, but for the remote system.

−d Increase the quantity of debugging output displayed during a test (possibly at the expense of performance).

−f GMKgmk

Change the units of measure for *_STREAM tests. Capital letters are powers of two, lowercase are powers of ten.

−F fill_file

Pre-fill the send buffers with data from the named file. This is intended to provide a means for avoiding buffers that are filled with data which is trivially easy to compress. A good choice for a file that should be present on any system is this manpage - netperf.man. Other files may be provided as part of the distribution.

−h Display a usage string, and exit.

−H remote_host,local_host

Set the hostname (or IP address) of the remote system. Passing a single name with no comma will only set remote_host and will leave selection of local IP address for the control connection to the stack. Specifying ",local_host" will only set local_host and will leave remote_host at the default.

−i max,min

Set the maximum and minimum number of iterations when trying to reach certain confidence levels.

−I lvl,[,intvl]

Specify the confidence level (either 95 or 99 - 99 is the default) and the width of the confidence interval as a percentage (default 10)

−l testlen

Specify the length of the test (default 10 seconds). A negative value sets the number of request/response transactions, or the number of bytes for a stream test.

−n numcpus
Specify the number of CPU's in the system on those systems for which netperf has no way to find the number of CPU's programatically.

−o sizespec
Set an offset from the alignment specified with -a.

−O sizespec
As -o, but for the remote system.

−p portnum,locport
Connect to a listening on the specified port, rather than using /etc/services. If ",locport" is specified the control connection will be established from that local port number. Specifying a single port number with no comma will specify only the remote port number and will leave local port number selection to the stack.

−P 0|1 Show (1) or suppress (0) the test banner.

−t testname
Specify the test to perform. Valid testnames are (but not always compiled-in):
TCP_STREAM
TCP_MAERTS
TCP_RR
TCP_CRR
UDP_STREAM
UDP_RR
TCPIPV6_STREAM
TCPIPV6_RR
TCPIPV6_CRR
UDPIPV6_STREAM
UDPIPV6_RR
DLCO_STREAM
DLCO_RR
DLCL_STREAM
DLCL_RR
STREAM_STREAM
STREAM_RR
DG_STREAM
DG_RR
LOC_CPU
REM_CPU

−v verbosity
Set the verbosity level for the test (only with -P).

−V Enable the copy-avoidance features (HP-UX 9.0 and later only).

TEST SPECIFIC OPTIONS

−h Display a usage string based on the test name set with -t, and exit.

Please consult the netperf manual *Netperf: A Network Performance Benchmark* (netperf.ps) for more information. Or you can join and mail to netperf-talk@netperf.org.

BUGS

There is a fairly large list of known defects and misfeatures in the manual. If you think you have found a bug, please send email to Rick Jones <raj@cup.hp.com>.

SEE ALSO

Netperf: A Network Performance Benchmark
http://www.netperf.org/

AUTHORS

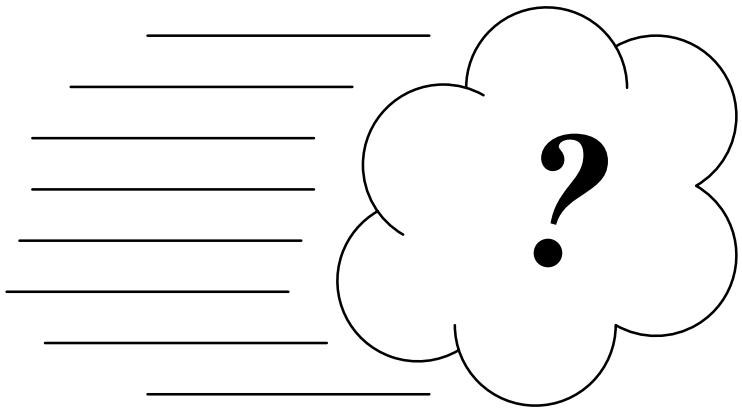
HP Information Networks Division - Networking Performance Team.
Rick Jones <raj@cup.hp.com>
Karen Choy HP IND
Dave Shield <daves@csc.liv.ac.uk> (man pages)
Others too numerous to mention here - see the ACKNWLDGMNTS fi le

Netperf:

A Network Performance Benchmark

Revision 2.1

Information Networks Division
Hewlett−Packard Company
February 15, 1996



Section 0. The Legal Stuff

Section 1. Introduction

Netperf is a benchmark that can be used to measure various aspects of networking performance. Its primary focus is on bulk data transfer and request/response performance using either TCP or UDP and the Berkeley Sockets interface. There are optional tests available to measure the performance of DLPI, Unix Domain Sockets, the Fore ATM API and the HP HiPPI LLA interface.

This tool is maintained and informally supported by the IND Networking Performance Team. It is **NOT** supported via any of the normal Hewlett−Packard support channels. You are free to make enhancements and modifications to this tool.

This document is organized (loosely) into several sections as follows:

- Section 1. is what you are reading right now.
- Section 2. describes how to get the netperf bits and how to set−up your system to run netperf. It also describes a simple way to verify that the installation has been successful.
- Section 3. describes the design of netperf.
- Section 4. describes netperf's bulk data transfer tests and their command line options.
- Section 5. describes netperf's request−response tests and their command options.
- Section 6. describes some of the supporting test types of netperf and their command line options.
- Section 7. provides a description of the global command−line options for netperf.
- Section 8. provides some examples of netperf usage.
- Section 9. lists the changes and fixes in this revision of netperf.
- Section 10. lists several known problems with this revision of netperf.
- Section 11. provides some troubleshooting assistance.

We thank you in advance for your comments, and hope that you find this tool useful.

The maintainers of netperf.

"How fast is it? It's so fast, that ..." ;−)

Conventions and Definitions

You may not be familiar with some of the conventions and definitions used by this document. Generally, items of particular importance, command line options, and commands will be in **boldface** type. Filenames and command line items requiring user substitution will appear in *italicized* type.

A *sizespec* is a one or two item list passed with a command line option that can set the value of one or two netperf parameters. If you wish to set both parameters to separate values, items should be separated by a comma − Eg "parm1,parm2". If you wish to set the first parameter without altering the value of the second, you should follow the first item with a comma − Eg "parm1,". Likewise, precede the item with a comma if you wish to set only the second parameter − Eg ",parm2". An item without a comma will set both parameters. This last mode is the one most frequently used.

Netperf has two types of command line options. The first are global command line options. They are essentially any option that is not tied to a particular test, or group of tests. An example of a global command line option is the test type. The second options are test specific options. These are options which are only applicable to a particular test. An example of a test specific option would be the send socket buffer size for a TCP_STREAM test. Global command line options are specified first, test specific second. They must be separated from each other by a "−−" (two dashes). If you wish to give test specific options only, they must be preceded by "−−". (EG ./netperf −− −m 1024)

Section 2. Installing Netperf

Netperf primary form of distribution is source code. This is to allow installation on systems other than those to which the authors have access and thus the ability to create binaries. There are two ways to install netperf. The first runs the netperf server program, netserver, as a child of inetd, which requires that the installer of netperf be able to edit the files */etc/services* and */etc/inetd.conf* (or their equivalent). The second is to run netserver as a standalone daemon. This second method does not require edit capabilities on */etc/services* and */etc/inetd.conf*. but does mean that you must remember to run the netserver program explicitly. The second method is required for Windows NT.

This manual assumes that those wishing to measure networking performance already know how to use anonymous FTP.

Getting the netperf bits from the Internet

For those people connected to the Internet, netperf is available via WWW. If you are not connected to the Internet such that you can use WWW, then you may be able to retrieve netperf via FTP or an FTP mail server. If all else fails, you can send email to Netperf Request <netperf−request@netperf.cup.hp.com>.

If you have a WWW browser, you can retrieve netperf via the Netperf Page. It is located at The Netperf Page . Follow the links from that page.

Netperf source bits are also available via anonymous FTP from **ftp.cup.hp.com** in the directory *dist/networking/benchmarks*. You should use **binary mode** transfers when bringing over the bits as you will be grabbing the latest copy of this document along with the netperf C source files.

NOTE: Older versions of netperf were available via anonymous FTP from **col.hp.com** under the directory *dist/networking/benchmarks/*. Other servers on the Internet may have copies. The “primary” place to go for netperf bits is ftp.cup.hp.com.

While the netperf source bits can be placed anywhere on the system, this manual will assume that the source bits are placed in the directory */opt/netperf/src*. Previous revisions of netperf were assumed to be placed in */usr/etc/net_perf/src*, but this has been changed to better emphasize that netperf is not an official Hewlett−Packard product. You are free to place netperf wherever you like, provided that you make the necessary modifications to the scripts.

Installing the bits

Once you have placed the netperf source bits onto the system, it is necessary to compile them and perform some editing tasks. This section assumes that you have elected to install the benchmark server, netserver, as a child of inetd.

The netperf distribution includes a makefile which assumes the existence of the directory */opt/netperf*. If you do not wish to have netperf installed there, it will be necessary for you to edit the makefile. To assist in this process, obvious markers have been placed in the makefile to indicate what must be changed. Also, some systems require different compile switches and libraries. For those systems where the requirements were known to the authors, comments have been added to the makefile.

Once the makefile is customized as needed, simply enter the command:

```
$ make install
```

from within the netperf source directory. The netperf executables will be compiled and copied into */opt/netperf* or the location specified in the makefile. Make will also copy the sample script files into the same place and verify that they are set to be executable.

If you do not have root access, or do not wish to install netperf as a child of inetd, skip to the subsection titled "Running netserver as a standalone Daemon."

Now that the executables have been created, it is necessary to edit the */etc/services* and */etc/inetd.conf* files. If you have decided to keep the netperf executables someplace other than */opt/netperf*, alter these lines accordingly. This editing step generally requires root access.

Add this line to the */etc/services* file:

```
netperf         12865/tcp
```

Then add this line to the */etc/inetd.conf* file:

```
netperf stream tcp nowait root /opt/netperf/netserver netserver
```

Running as root is probably no longer required (It was used on older versions of netperf which read from /dev/kmem) so if you are uncomfortable running netserver as root, you can pick another id. Once the files have been edited, it is necessary to have inetd re-configure itself. On an HP-UX system, this is accomplished with the command:

```
$ /etc/inetd -c
```

On some systems it is possible to get inetd to re-configure itself by sending it a SIGHUP with the kill(2) command:

```
$ kill -HUP <pid of inetd>
```

On other systems it might be necessary to kill and re-start the inet daemon. At this point, root access is no longer needed and you can proceed to the verification step.

Verifying the bits

To verify the installation of netperf, simply execute the command

```
/opt/netperf/netperf
```

A TCP_STREAM test of 10 seconds duration should be performed over the loopback interface.

Running netserver as a standalone Daemon

If you cannot install netperf as a child of inetd, you can run the netserver as a standalone daemon. Simply execute netserver with the "-p <port number>" option and it will happily start accepting requests on the port number you specify. If you specify a port number other than the normal netperf port number, you should remember to also specify "-p <portnum>" as a global command line option of netperf.

Final Customization

The scripts provided with the netperf distribution are written with the assumption that netperf is installed in */opt/netperf*. If you have decided to install netperf in a different location, you will need to edit each of the script files and alter this line:

NETHOME=/opt/netperf

or one like it, to be something like:

NETHOME=/my/netperf/location

Section 3. The Design of Netperf

Design Basics

Netperf is designed around the basic client−server model. There are two executables − netperf and netserver. Generally you will only execute the netperf program − the netserver program will be invoked by the other system's inetd.

When you execute netperf, the first thing that will happen is the establishment of a control connection to the remote system. This connection will be used to pass test configuration information and results to and from the remote system. Regardless of the type of test being run, the control connection will be a TCP connection using BSD sockets.

Once the control connection is up and the configuration information has been passed, a separate connection will be opened for the measurement itself using the APIs and protocols appropriate for the test. The test will be performed, and the results will be displayed.

Netperf places no traffic on the control connection while a test is in progress. Certain TCP options, such as SO_KEEPALIVE, if set as your system's default, may put packets out on the control connection.

CPU Utilization

CPU utilization is a frequently requested metric of networking performance. Unfortunately, it can also be one of the most difficult metrics to measure accurately. Netperf is designed to use one of several (perhaps platform dependent) CPU utilization measurement schemes. Depending on the CPU utilization measurement technique used, a unique single−letter code will be included in the CPU portion of the test banner for both the local and remote systems.

The default CPU measurement technique is based on the use of "loopers" which will sit in tight little loops consuming any CPU left over by the networking. This method is not without its added overhead, but wherever possible, care has been taken to keep that overhead to a minimum. If you would like to get an estimate of the overhead, run one test with CPU utilization, and one test without, and compare the throughputs. Use of loopers in measuring CPU utilization is indicated by the letter code "L."

NOTE: For accurate CPU utilization on MP systems, it is *crucial* that netperf and netserver know the number of processors on the system. For some systems (HP−UX) this can be determined programmatically. Other systems require the use of the "−n" global command line argument.

HP−UX 10.X offers a zero additional overhead, very accurate CPU utilization mechanism based on the pstat() system call. If you are compiling on HP−UX 10, you should replace the "−DUSE_LOOPER" in the makefile with "−DUSE_PSTAT" and recompile. When this method is being used, the letter code "I" will be displayed.

Other codes may be included in later versions of netperf. When the CPU utilization mechanism is unknown, either a "U" or a "?" will be displayed.

Great care should be exercised when looking at CPU utilization. Be certain you are familiar with the technique being used, and its implications. For example, a mechanism that is based solely on CPU charged to the netperf (netserver) process alone will likely under−report the real CPU utilization SIGNIFICANTLY. Much network processing takes place away from the user process context. Caveat Benchmarker!

Section 4. Using Netperf to measure bulk data transfer performance

The most common use of netperf is measuring bulk data transfer performance. This is also referred to as “stream” or “unidirectional stream” performance. Essentially, these tests will measure how fast one system can send data to another and/or how fast that other system can receive it.

TCP Stream Performance

The TCP stream performance test is the default test type for the netperf program. The simplest test is performed by entering the command:

/opt/netperf/netperf −H *remotehost*

which will perform a 10 second test between the local system and the system identified by *remotehost*. The socket buffers on either end will be sized according to the systems' default and all TCP options (e.g. TCP_NODELAY) will be at their default settings.

To assist in measuring TCP stream performance, two script files are provided with the netperf distribution. They are *tcp_stream_script* and *tcp_range_script*. *Tcp_stream_script* will invoke netperf based on the setting of script variables controlling socket and send sizes. *Tcp_range_script* will perform a similar set of tests, with the difference being that where *tcp_stream_script* tests specific datapoints, *tcp_range_script* will perform tests at points within a specified range.

If you would like to perform tests other than those done by the scripts, you can invoke netperf manually. Some of the options you will likely want to experiment with are:

−s *sizespec*	which will set the local send and receive socket buffer sizes to the value(s) specified. [Default: system default socket buffer sizes]
−S *sizespec*	which behaves just like −s but for the remote system
−m *value*	set the local send size to *value* bytes. [Default: local socket buffer size]
−M *value*	which behaves like −m, setting the receive size for the remote system. [Default: remote receive socket buffer size]
−l *value*	set the test length to *value* seconds when value is > 0 and to $\|value\|$ bytes when value is < 0
−D	set the TCP_NODELAY option to true on both systems

This is not a complete list of options that can affect TCP stream performance, but it does cover those options that are used most often. A complete list of netperf options can be found in Section 7.

XTI TCP Stream Performance

The XTI TCP stream performance test quite similar to the TCP_STREAM test. XTI requires a device file be opened – as the device file is placed in different locations on different systems, it generally must be specified. The simplest XTI TCP stream test on HP–UX is performed by entering the command:

/opt/netperf/netperf –H *remotehost* **–t** XTI_TCP_STREAM **–– –X** /dev/inet_cots

which will perform a 10 second test between the local system and the system identified by *remotehost*. The socket buffers on either end will be sized according to the systems' default and all TCP options (e.g. TCP_NODELAY) will be at their default settings.

The test parameters for an XTI_TCP_STREAM test are the same as for a TCP_STREAM test with the addition of:

–X *devspec* set the local/remote XTI device file name from *devspec*.

UDP Stream Performance

A UDP stream performance test is very similar to a TCP stream test. One difference is that the send size cannot be larger than the smaller of the local and remote socket buffer sizes. What this means is that you must make certain that when you specify the –m option, you use a value that is less than or equal to the socket buffer sizes (–s and –S). Also, since the UDP Stream test is not the default test, the –t *testname* option must be specified, with the testname set to UDP_STREAM. So, a simple UDP stream test command might look something like this:

$ **/opt/netperf/netperf –H** *remotehost* **–t** UDP_STREAM **–– –m** 1024

There is a script provided that performs various UDP stream performance tests. It is called *udp_stream_script*. As with TCP stream performance, you can use the script provided, or perform tests yourself to get datapoints not covered by the script.

NOTE: UDP is an unreliable protocol. It is important that you examine the results carefully as the reported send rate can be much higher than the actual receive rate. Great care should be taken when reporting UDP_STREAM test results to make sure they are not misleading. For example, one should **always** report both send and receive rates **together** for a UDP_STREAM test. If you are going to report a single number, you should report the receive rate.

NOTE: If you would like to "pace" the send rate of the UDP_STREAM test, add a –DINTERVALS to the makefile, do a "make clean" and re–compile. You can then use the –b and –w global options to set the burst size (sends) and wait time (milliseconds) respectively.

XTI UDP Stream Performance

The XTI UDP stream performance test quite similar to the UDP_STREAM test. XTI requires a device file be opened. As the device file is placed in different locations on different systems, it generally must be specified. The simplest XTI UDP stream test on HP–UX is performed by entering the command:

/opt/netperf/netperf –H *remotehost* **–t** XTI_UDP_STREAM **–– –X** /dev/inet_clts

The test parameters for an XTI_UDP_STREAM test are the same as for a UDP_STREAM test with the addition of:

−X *devspec* set the local/remote XTI device file name from *devspec*.

NOTE: UDP is an unreliable protocol. It is important that you examine the results carefully as the reported send rate can be much higher than the actual receive rate. Great care should be taken when reporting XTI_UDP_STREAM test results to make sure they are not misleading. For example, one should **always** report both send and receive rates **together** for a XTI_UDP_STREAM test. If you are going to report a single number, you should report the receive rate.

NOTE: If you would like to "pace" the send rate of the XTI_UDP_STREAM test, add a −DINTERVALS to the makefile, do a "make clean" and re−compile. You can then use the −b and −w global options to set the burst size (sends) and wait time (milliseconds) respectively.

DLPI Connection Oriented Stream Performance

NOTE: DLPI tests are not compiled−in by default with netperf. If you wish to measure performance over DLPI, you will need to add a −DDO_DLPI to the makefile and perhaps add to the "LIBS=" and re−compile netperf and netserver.

A DLPI Connection Oriented Stream test (DLCO_STREAM) looks very similar to a TCP Stream test − they both use reliable, connection oriented protocols. The DLPI test differs from the TCP test in that the message size must always be less than or equal to the local interface's MTU − DLPI does not provide TCP−style segmentation and reassembly.

The simplest DLPI Connection Oriented Stream test would look something like this:

$ **/opt/netperf/netperf −H** *remotehost* **−t** DLCO_STREAM −− **−m** 1024

Here are some of the DLPI−specific command line options:

−D *devspec*	specify the local and/or remote DLPI device file name(s) (fully−qualified). Syntax is the same as that of a *sizespec*.
−m *value*	specify the send size, in bytes, for the local system. This must be less than or equal to the link MTU.
−M *value*	which behaves like −m, setting the receive size for the remote system.
−p *ppaspec*	set the local and/or remote DLPI PPA(s). Syntax is the same as that of a *sizespec*.
−r *value*	specify the request size, in bytes, for the test.
−R *value*	specify the response size, in bytes, for the test.
−s *value*	specify the 802.2 SAP for the test. This should not conflict with any assigned SAP's.
−w *sizespec*	specify the local send/recv window sizes in frames (where available).
−W *sizespec*	specify the remote send/recv window sizes in frames (where available).

DLPI Connectionless Stream

NOTE: DLPI tests are not compiled−in by default with netperf. If you wish to measure performance over DLPI, you will need to add a −DDO_DLPI to the makefile and perhaps add to the "LIBS=" and re−compile netperf and netserver.

A DLPI Connectionless Stream test (DLCL_STREAM) is analogous to a UDP_STREAM test. They both make use of unreliable, connectionless transports. The DLPI test differs from the UDP test in that the message size must always be less than or equal to the link MTU − DLPI does not provide IP−like segmentation and reassembly functionality, and the netperf benchmark does not presume to provide one.

The simplest DLPI Connectionless Stream test command line would look something like this:

$ **/opt/netperf/netperf −H** *remotehost* **−t** DLCL_STREAM −− **−m** 1024

Here are some of the DLPI−specific command line options for the DLCL_STREAM test:

−D *devspec*	specify the local and/or remote DLPI device file name(s) (fully−qualified). Syntax is the same as that of a *sizespec*.
−m *value*	specify the send size, in bytes, for the local system. This must be less than or equal to the link MTU.
−M *value*	which behaves like −m, setting the receive size for the remote system.
−p *ppaspec*	set the local and/or remote DLPI PPA(s). Syntax is the same as that of a *sizespec*.
−s *value*	specify the 802.2 SAP for the test. This should not conflict with any assigned SAP's.
−w *sizespec*	specify the local send/recv window sizes in frames (where available).
−W *sizespec*	specify the remote send/recv window sizes in frames (where available).

Unix Domain Stream Sockets

NOTE: Unix Domain Socket tests are not compiled into netperf by default. If you wish to measure the performance of Unix Domain Sockets, you must recompile netperf and netserver with −DDO_UNIX added to the makefile.

A Unix Domain Stream Socket Stream test (STREAM_STREAM) is very much like a TCP_STREAM test.

The Simplest Unix Domain Stream Socket Stream test command line would look something like this:

$ **/opt/netperf/netperf −t** STREAM_STREAM

The −H global command line Option is not valid for a Unix Domain Socket test and should not be specified.

Here are some of the Unix Domain−specific command line options for the STREAM_STREAM test:

−m *value*	set the local send size to *value* bytes. [Default: local socket buffer size]
−M *value*	which behaves like −m, setting the receive size for the remote system. [Default: remote receive socket buffer size]
−p *dirspec*	set the directory where pipes will be created. [Default: system default for the tempnam() call]
−s *sizespec*	which will set the local send and receive socket buffer sizes to the value(s) specified. [Default: system default socket buffer sizes]
−S *sizespec*	which behaves just like −s but for the remote system

Unix Domain Datagram Sockets

NOTE: Unix Domain Socket tests are not compiled into netperf by default. If you wish to measure the performance of Unix Domain Sockets, you must recompile netperf and netserver with −DDO_UNIX added to the makefile.

A Unix Domain Datagram Socket Stream test (DG_STREAM) is very much like a TCP_STREAM test except that message boundaries are preserved.

The Simplest Unix Domain Datagram Socket Stream test command line would look something like this:

$ **/opt/netperf/netperf −t** DG_STREAM

The −H global command line option is not valid for a Unix Domain Socket test and should not be specified. Here are some of the test specific command line options available in a DG_STREAM test.

−m *value*	set the local send size to *value* bytes. [Default: local socket buffer size]
−M *value*	which behaves like −m, setting the receive size for the remote system. [Default: remote receive socket buffer size]
−p *dirspec*	set the directory where pipes will be created. [Default: system default for the tempnam() call]
−s *sizespec*	which will set the local send and receive socket buffer sizes to the value(s) specified. [Default: system default socket buffer sizes]
−S *sizespec*	which behaves just like −s but for the remote system

Fore ATM API Stream

NOTE: Fore ATM API tests are not compiled into netperf by default. If you wish to measure the performance of connections over the Fore ATM API, you must recompile netperf and netserver with −DDO_FORE added to the makefile.

A Fore ATM API Stream test (FORE_STREAM) is very much like a UDP_STREAM test.

NOTE: The Fore ATM API exports an unreliable protocol. It is important that you examine the results carefully as the reported send rate can be much higher than the actual receive rate. Great care should be taken when reporting FORE_STREAM test results to make sure they are not misleading. For example, one should **always** report both send and receive rates **together** for a FORE_STREAM test. If you are going to report a single number, you should report the receive rate.

The simplest Fore ATM API Stream test command line would look something like this:

$ **/opt/netperf/netperf −t** FORE_STREAM **−H** *remotehost*

Here are some of the test specific command line options applicable to a FORE_STREAM test.

−a *AAL*	use the ATM Adaptation Layer number aal to encapsulate packets. Specifying 3 or 4 will yield AAL3/4, and 5 will yield AAL5. [Default: 5 −> AAL5]
−b *sizespec*	set the mean burst target and/or minimum in units of kilobit packets. The first value is target and the second is minimum. [Default: 0,0]
−d *devspec*	set the name of the ATM device file to be opened. [Default: /dev/atm]
−m *value*	set the local send size to *value* bytes. This must not be larger than the ATM MTU. [Default: ATM MTU]
−M *value*	which behaves like −m, setting the receive size for the remote system. [Default: ATM MTU]
−p *sizespec*	set the peak bandwidth target and/or minimum in units of kilobits/s. The first value is target and the second it minimum. [Default: 0,0 −> network assigned]
−P *sizespec*	set the mean bandwidth target and/or minimum in units of kilobits/s. The first value is target and the second is minimum. [Default: 0,0 −> network assigned]

Section 5. Using Netperf to measure request/response performance

Request/response performance is the second area that can be investigated with netperf. Generally speaking, netperf request/response performance is quoted as "transactions/s" for a given request and response size. A transaction is defined as the exchange of a single request and a single response. From a transaction rate, one can infer one way and round−trip average latency.

TCP Request/Response Performance

The TCP request/response test can be invoked with netperf though the use of the −t option with an argument of TCP_RR. So, a "default" request/response command would look something like this:

$ **/opt/netperf/netperf −H** *remotehost* **−t** TCP_RR

and will use the system default socket buffer sizes, a default request size of 1 byte, and a default response size of 1 byte.

As with the stream performance tests, a script is available to assist you in generating TCP request/response performance numbers. It is called *tcp_rr_script*. However, if you should need to generate numbers at points of you own choosing, these command line options will be of use:

−r	*sizespec*	set the request and/or response sizes based on *sizespec*.
−l	*value*	set the test duration based on *value*. For *value* > 0, test duration will be *value* seconds. Otherwise, test duration will be |*value*| transactions.
−s	*sizespec*	which will set the local send and receive socket buffer sizes to the value(s) specified. [Default: system default socket buffer sizes]
−S	*sizespec*	which behaves just like −s but for the remote system
−D		set the TCP_NODELAY option to true on both systems

The request and response sizes will be the buffer sizes posted to send and receive. The −m and −M options are not meaningful for a TCP_RR test.. As TCP is a stream protocol and not a message protocol, it is necessary to loop on receives until the entire message is delivered. The buffer pointer passed to the first receive for an individual transaction will be aligned and offset as requested by the user. It will be incremented by the number of bytes received each time until the entire request/response is received. The buffer pointer will be re−aligned and offset for the next transaction.

TCP Connect/Request/Response

The TCP_CRR test is a test which mimics the http protocol used by most web servers. Instead of simply measuring the performance of request/response in the same connection, it establishes a new connection for each request/response pair. The test−specific parameters are the same as the TCP_RR test, with one addition:

−p	*max*[,*min*]	set min/max port numbers used by the client side.

It is important that this test run for a reasonable length of time — at least two minutes. This is related to the behavior of various TCP implementations. If you run the test for shorter periods of time, the results could be higher than seen in a steady-state condition. So, a good TCP_CRR command line to simulate a web-server might look like:

$ **/opt/netperf/netperf -t** TCP_CRR **-l** 120 **-H** *remotehost* **-- -r** 32,1024

XTI TCP Request/Response Performance

The XTI TCP request/response test can be invoked with netperf though the use of the -t option with an argument of XTI_TCP_RR. Not all systems put the requisite device files in the same location, so, a "default" request/response command on HP-UX would look something like this:

$ **/opt/netperf/netperf -H** *remotehost* **-t** XTI_TCP_RR **-- -X** /dev/inet_cots

and will use the system default socket buffer sizes, a default request size of 1 byte, and a default response size of 1 byte.

The command-line options for the XTI_TCP_RR test are the same as the TCP_RR test, with the following additions:

-X *devspec* set the local/remote XTI device file name from *devspec*.

The request and response sizes will be the buffer sizes posted to send and receive. As TCP is a stream protocol and not a message protocol, it is necessary to loop on receives until the entire message is delivered. The buffer pointer passed to the first receive for an individual transaction will be aligned and offset as requested by the user. It will be incremented by the number of bytes received each time until the entire request/response is received. The buffer pointer will be re-aligned and offset for the next transaction.

UDP Request/Response Performance

UDP request/response performance works just like TCP request/response performance. All the options available there are present here with the exception of the -D option; TCP_NODELAY has no meaning for a UDP test. To invoke a UDP request/response test, use an argument of UDP_RR with the -t option to produce a command like something like this:

$ **/opt/netperf/netperf -H** *remotehost* **-t** UDP_RR

Again, a script is provided which will generate results for some of the more common datapoints. It is named *udp_rr_script*.

XTI UDP Request/Response Performance

The XTI UDP request/response test can be invoked with netperf though the use of the -t option with an argument of XTI_UDP_RR. Not all systems put the requisite device files in the same location, so, a "default" request/response command on HP-UX would look something like this:

$ **/opt/netperf/netperf -H** *remotehost* **-t** XTI_UDP_RR **-- -X** /dev/inet_clts

and will use the system default socket buffer sizes, a default request size of 1 byte, and a default response size of 1 byte.

The command–line options for the XTI_UDP_RR test are the same as the UDP_RR test, with the following additions:

–X *devspec*	set the local/remote XTI device file name from *devspec*.

The request and response sizes will be the buffer sizes posted to send and receive.

DLPI Connection Oriented Request/Response Performance

NOTE: DLPI tests are not compiled into netperf by default. If you wish to measure the performance of DLPI, you must recompile netperf and netserver with –DDO_DLPI added to the makefile.

A DLPI Connection Oriented Request/Response test (DLCO_RR) looks much the same as any other request/response test. It performs a request/response test over a reliable connection. As with the other DLPI tests, there is no segmentation and reassembly, so all request and/or response sizes must be less than or equal to the link MTU.

A simple DLCO_RR test invocation would look something like this:

$ **/opt/netperf/netperf –H** *remotehost* **–t** DLCO_RR

Here are some of the DLPI–specific command line options:

–D *devspec*	specify the local and/or remote DLPI device file name(s) (fully–qualified). Syntax is the same as that of a *sizespec*.
–p *ppaspec*	set the local and/or remote DLPI PPA(s). Syntax is the same as that of a *sizespec*.
–r *sizespec*	specify the request and/or response sizes, in bytes, for the test.
–s *value*	specify the 802.2 SAP for the test. This should not conflict with any assigned SAP's.
–w *sizespec*	specify the local send/recv window sizes in frames (where available).
–W *sizespec*	specify the remote send/recv window sizes in frames (where available).

DLPI Connectionless Request/Response Performance

NOTE: DLPI tests are not compiled into netperf by default. If you wish to measure the performance of DLPI, you must recompile netperf and netserver with –DDO_DLPI added to the makefile.

A DLPI Connectionless Request/Response test (DLCL_RR) looks much the same as any other request/response test. It performs a request/response test over an unreliable connection. However, netperf does not have any sort of retransmission mechanism, so packet loss with this test will result in dramatically lowered performance results. As with the other DLPI tests, there is no segmentation and reassembly, so all request and/or response sizes must be less than or equal to the link MTU.

A simple DLCL_RR test invocation would look something like this:

$ **/opt/netperf/netperf –H** *remotehost* **–t** DLCL_RR

Here are some of the DLPI–specific command line options:

−D *devspec*	specify the local and/or remote DLPI device file name(s) (fully−qualified). Syntax is the same as that of a *sizespec*.
−p *ppaspec*	set the local and/or remote DLPI PPA(s). Syntax is the same as that of a *sizespec*.
−r *sizespec*	specify the request and/or response sizes, in bytes, for the test.
−s *value*	specify the 802.2 SAP for the test. This should not conflict with any assigned SAP's.
−w *sizespec*	specify the local send/recv window sizes in frames (where available).
−W *sizespec*	specify the remote send/recv window sizes in frames (where available).

Unix Domain Stream Socket Request/Response Performance

NOTE: Unix Domain Socket tests are not compiled into netperf by default. If you wish to measure the performance of Unix Domain Sockets, you must recompile netperf and netserver with −DDO_UNIX added to the makefile.

A Unix Domain Stream Socket Request/Response test (STREAM_RR) is very much like a TCP_RR test.

The STREAM_RR test command line would look something like this:

$ **/opt/netperf/netperf −t** STREAM_RR

The −H global command line option is not valid for a Unix Domain Socket test and should not be specified.

Here are some of the Unix Domain−specific command line options for the STREAM_STREAM test:

−p *dirspec*	set the directory where pipes will be created. [Default: system default for the tempnam() call]
−r *sizespec*	which will set the request and response sizes to the value(s) specified. [Default: 1 byte]

Unix Domain Datagram Socket Request/Response Performance

NOTE: Unix Domain Socket tests are not compiled into netperf by default. If you wish to measure the performance of Unix Domain Sockets, you must recompile netperf and netserver with −DDO_UNIX added to the makefile.

The Simplest Unix Domain Datagram Socket Request/Response (DG_RR) test command line would look something like this:

$ **/opt/netperf/netperf −t** DG_STREAM

The −H global command line option is not valid for a Unix Domain Socket test and should not be specified. Here are some of the test specific command line options available in a DG_STREAM test.

−p *dirspec* set the directory where pipes will be created. [Default: system default for the tempnam() call]

−r *sizespec* set the request and/or response sizes to the value(s) specified. [Default: 1 byte]

Fore ATM API Request/Response Performance

NOTE: Fore ATM API tests are not compiled into netperf by default. If you wish to measure the performance of connections over the Fore ATM API, you must recompile netperf and netserver with −DDO_FORE added to the makefile.

A Fore ATM API Request/Response test (FORE_RR) is very much like a UDP_RR test.

The simplest FORE_RR test command line would look something like this:

$ **/opt/netperf/netperf −t** FORE_RR **−H** *remotehost*

Here are some of the test specific command line options applicable to a FORE_STREAM test.

−a *aal* use the ATM Adaptation Layer number aal to encapsulate packets. Specifying 3 or 4 will yield AAL3/4, and 5 will yield AAL5. [Default: 5 −> AAL5]

−b *sizespec* set the mean burst target and/or minimum in units of kilobit packets. The first value is target and the second is minimum. [Default: 0,0]

−d *devspec* set the name of the ATM device file to be opened. [Default: /dev/atm]

−p *sizespec* set the peak bandwidth target and/or minimum in units of kilobits/s. The first value is target and the second it minimum. [Default: 0,0 −> network assigned]

−P *sizespec* set the mean bandwidth target and/or minimum in units of kilobits/s. The first value is target and the second is minimum. [Default: 0,0 −> network assigned]

−r *sizespec* set the request and/or response sizes to the values specified [Default: 1 byte]

Section 6. Other Netperf tests

Apart from the usual performance tests, netperf contains some tests that can be used to streamline measurements. These tests range from CPU rate calibration (present) to host identification (future enhancement).

CPU rate calibration

NOTE: Previous revisions of the manual may have discussed the HP kernel idle counter. With the release of HP−UX 10.0, this mechanism has been replaced with an equally accurate one based on information returned from the pstat() system call. This accurate pstat−based mechanism is *not* available in HP−UX 9.X, nor in non−HP operating systems. For those systems, a CPU utilization measurement based on the use of "loopers" is employed. Each requires calibration.

In the context of netperf, a CPU rate is expressed not in clock frequencies, MIPS or MFLOPS, but simply how fast the system can count. There are two CPU rate calibrations tests. The first measures and displays the CPU rate for the local system. It is called LOC_CPU. The second test, REM_CPU, is exactly the same, except that it works on the system specified with the −H command line option.

In and of themselves, these two tests are only arcanely interesting. However, they can be used to greatly speed−up test scripts. Remember that for CPU measurements, it is necessary to "calibrate" the CPU or determine how fast it can count. This process takes at least forty (40) seconds for the local system and forty (40) seconds for the remote system. One can save the results of the CPU tests in shell variables and then use them as arguments to the −c and −C command line options. Passing−in a rate with the −c or −C option tells netperf that you already know the CPU rate, so it can skip the calibration steps. For example, the following Unix shell fragment will determine the local CPU rate and use that for subsequent tests:

$ LOC_RATE=‘**/opt/netperf/netperf −t** LOC_CPU‘

$ **/opt/netperf/netperf −H** *somehost* **−c** $LOC_RATE

You should remember that CPU rates will vary from system to system. Generally, the best trade−off between time and accuracy is to perform the calibrations once in each script or session. The default scripts provided will use the LOC_CPU and REM_CPU tests to reduce the time overhead of CPU calibration.

NOTE: For accurate CPU utilization on MP systems, it is *crucial* that netperf and netserver know the number of processors on the system. For some systems (HP−UX) this can be determined programmatically. Other systems require the use of the "−n" global command line argument. The authors are always looking for supported calls to find the number of active processors in a system.

Section 7. Netperf Command-line Options Reference

This section describes each of the global command-line options available in the netperf program. Essentially, it is an expanded version of the usage information displayed by netperf when invoked with the -h option in global command line option area.

Command-line Options Syntax

Revision 1.8 of netperf introduced enough new functionality to overrun the English alphabet for mnemonic command line option names. For this reason, command-line options were split in Revision 1.8. This split remains in Revision 1.9alpha. There are two types of netperf command-line options. They are "global" and "test-specific." Both types are entered on the same command line, but they must be separated by a "--" for correct parsing. Global command line options come first, followed by test-specific. If only test-specific options are to be specified, they must be preceded by "--" or the results will be undefined.

Global Options

-a sizespec
: This option allows you to alter the send and receive buffer alignments on the local system. Changing the alignment of the buffers can force the system to use different copying schemes, which can have a measurable impact on performance. If the page size for the system was 4096 bytes, and you wanted to pass page aligned buffers beginning on page boundaries, you could use "-a 4096". The units for this option are whole bytes. [Default: 8 bytes]

-A sizespec
: This option is identical to the -a option with the exception that the alignments are altered for the remote system.

-b size
: This option (-DINTERVALS compilation only) sets the size of a burst of packets in a _STREAM test. This can be used to "pace" the send rate when there is no flow-control provided by the protocol being measured.

-c [rate]
: This option will request CPU utilization and service demand calculations for the local system. If the optional rate parameter is specified, netperf will use that instead of calculating the rate itself. For more information on CPU utilization measurements with netperf, please consult Section 3. [Default: no CPU measurements]

-C [rate]
: This option is identical to the -c option with the exception that it requests CPU utilization for the remote system.

-d
: This option will increase the quantity of debugging output displayed during a test. If debugging is set high enough, it may have a measurable impact on performance. Debugging information for the local system (the one running netperf) is printed to stdout. Debugging information for the remote system (the one running netserver) is sent to the file /tmp/netperf.debug [Default: no debugging]

−f	GMKgmk	This option can be used to change the units of measure for stream tests. The "G", "M", and "K" arguments will set the output units to 2^{30}, 2^{20}, and 2^{10} bytes/s respectively. The "g", "m", and "k" arguments will set the output units to 10^9, 10^6, and 10^3 bits/s respectively. [Default: m − 10^6 bits/s)]
−h		This option causes netperf to display its usage string and exit.
−H	remotehost	This option sets the name of the remote system. It can be specified as either a hostname (e.g. foo.bar.baz) or an IP address (e.g. 1.2.3.4). [Default: localhost]
−l	testlen	With this option you can control the length of the test. If you specify a positive value for testlen, the test will run for that many seconds. If you specify a negative value, the test will run for that many transactions for a request/response test, or that many bytes for a stream test. Some tests can only be timed. [Default: 10 seconds]
−n	value	The value passed−in will be used as the number of CPU's in the system for the purposes of CPU utilization. This is required for MP systems where netperf cannot determine the number of processors programmatically. It is not needed on HP−UX 10.X. [Default: 1 processor]
−o	sizespec	The value passed with this option will be used as an offset from the alignment specified with the −a option. With this option you could, for example, pass buffers to the system that began 3 bytes after the beginning of a 4KB page (−a 4096 −o 3) [Default: 0 bytes]
−O	sizespec	This option behaves just like the −o option but on the remote system. It works in conjunction with the −A option. [Default: 0 bytes]
−p	portnum	You should use this option when the netserver program will be waiting at a port other than the default. This might be the case if you run netserver as a standalone process rather than a child of inetd.
−P	0|1	If you do not want the test banner to be displayed, then use this option with an argument of 0. An situation where this might be useful would be where you repeat the same test configuration several times and do not want the banners cluttering things up. [Default: 1 − display test banners]

−t testname
: You should use this option to specify the test you wish to perform. As of this writing, the valid testnames are TCP_STREAM, TCP_RR, UDP_STREAM, UDP_RR, DLCO_STREAM, DLCO_RR, DLCL_STREAM, DLCL_RR, STREAM_STREAM, STREAM_RR, DG_STREAM, DG_RR, FORE_STREAM, FORE_RR, HIPPI_STREAM, HIPPI_RR LOC_CPU, and REM_CPU. [Default: TCP_STREAM]

−v verbosity
: This option can be used to set the verbosity level for the test. It can be used in conjunction with the −P option. If the verbosity is set to 0, then only the result of the test will be displayed. If CPU utilization numbers are requested for the local system, the result of the test will be local service demand. If remote CPU utilization is requested, then the result will be remote service demand. Otherwise, the result will be the measured thruput.

−V
: This option will attempt to enable the copy−avoidance features of HP−UX 9.0 networking. [Default: no copy−avoidance attempted]

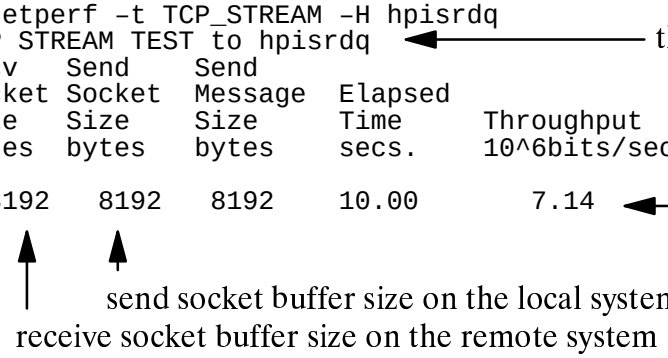
−w time
: This option (−DINTERVALS compilation only) will set the inter−burst time to time milliseconds. The actual wait time may differ depending on the resolution of timers on the system being measured.

Section 8. Netperf examples

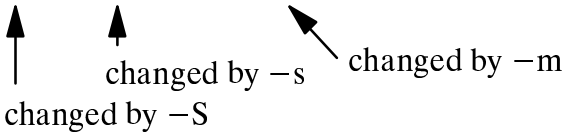
NOTE: These examples are from an older version of netperf and do not represent the split between global and test specific command line options.

The next few pages contain annotated screen dumps of example netperf runs. After examining these examples, you should have a fairly good idea of how to read the output of netperf and what effect certain options have on that output. First, TCP_STREAM tests.

```
$ netperf -t TCP_STREAM -H hpisrdq
TCP STREAM TEST to hpisrdq
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

  8192   8192   8192    10.00       7.14
```

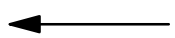



```
$ netperf -t TCP_STREAM -H hpisrdq -- -s 16384 -S 16K -m 1K
TCP STREAM TEST to hpisrdq
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 16384  16384   1024    10.01       7.32
```




```
$ netperf -t TCP_STREAM -H hpisrdq -P 0
  8192   8192   8192    10.00       8.07
```


```
$ netperf -t TCP_STREAM -H hpisrdq -P 0 -v 0
   7.05
```



This next set of examples is taken from some UDP_STREAM tests. You should notice right away that the output format is slightly different as there can be UDP sends that "fail" without there being a connection loss. Also, since every UDP datagram sent is not always "sent" or received, there are more statistics displayed – one line for local statistics and a second line for remote statistics.

```
$ ./netperf -t UDP_STREAM
UDP UNIDIRECTIONAL SEND TEST to localhost
Socket  Message  Elapsed      Messages
Size    Size     Time         Okay Errors   Throughput
bytes   bytes    secs            #      #   10^6bits/sec

  9216    9216   9.99        10500      0      77.47
  9360           9.99        10314             76.10
```

test type and dest

message size

successful calls to send

send performance

```
$ ./netperf -t UDP_STREAM -f K
UDP UNIDIRECTIONAL SEND TEST to localhost
Socket  Message  Elapsed      Messages
Size    Size     Time         Okay Errors   Throughput
bytes   bytes    secs            #      #   KBytes/sec

  9216    9216   10.00        9822      0    8839.06
  9360           10.00        9534           8579.88
```

receive perf

receive socket size on remote

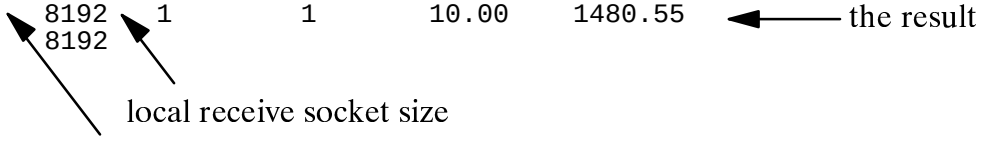
successful calls to recv

```
$ ./netperf -t UDP_STREAM -H hpindio -- -m 1472
UDP UNIDIRECTIONAL SEND TEST to hpindio
Socket  Message  Elapsed      Messages
Size    Size     Time         Okay Errors   Throughput
bytes   bytes    secs            #      #   10^6bits/sec

  9216    1472   10.00        7634  45525       8.99
  9360           10.00        7572              8.92
```

failed send calls (ENOBUFS)

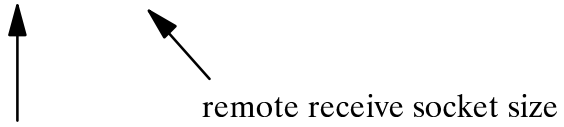
This third set of examples is taken from some *_RR tests. These tests use a two–line results format much like that of the UDP_STREAM tests. The only exception is that there are no provisions made for displaying lost packets, as there are not to be any.

```
$ ./netperf -t TCP_RR
TCP REQUEST/RESPONSE TEST to localhost
Local /Remote
Socket Size   Request  Resp.   Elapsed  Trans.
Send   Recv   Size     Size    Time     Rate
bytes  Bytes  bytes    bytes   secs.    per sec

8192   8192   1        1       10.00    1480.55
8192   8192
```



use 1024 byte requests and 256 bytes responses

```
$ ./netperf -t UDP_RR -- -r 1024,256
UDP REQUEST/RESPONSE TEST to localhost
Local /Remote
Socket Size   Request  Resp.   Elapsed  Trans.
Send   Recv   Size     Size    Time     Rate
bytes  Bytes  bytes    bytes   secs.    per sec

9216   9360   1024     256     10.00    1421.35
9216   9360
```



If you have compiled netperf with –DHISTOGRAM, it is possible to get a histogram of individual response times for a request/response test. To enable the histogram code, you need to specify a verbosity level of two or more with the "–v" global command.

```
$ ./netperf –l 60 –t TCP_RR –H hpindio –v 2
TCP REQUEST/RESPONSE TEST to hpindio : histogram
Local /Remote
Socket Size   Request  Resp.   Elapsed  Trans.
Send   Recv   Size     Size    Time     Rate
bytes  Bytes  bytes    bytes   secs.    per sec

8192   8192   1        1       60.00     779.55
8192   8192
Alignment      Offset
Local  Remote  Local  Remote
Send   Recv    Send   Recv
    8      0       0      0
Histogram of request/response times
TENTH_MSEC    :    0:    0:    0:    0:    0:    0:    0:    0:   0:    0
UNIT_MSEC     :    0: 45771:  582:  141:   42:   48:   48:   38:   24:   10
TEN_MSEC      :    0:   51:    8:    1:    1:    0:    0:    1:    1:    0
HUNDRED_MSEC  :    0:    2:    0:    0:    0:    1:    0:    0:    0:    0
UNIT_SEC      :    0:    0:    0:    0:    0:    0:    0:    0:    0:    0
TEN_SEC       :    0:    0:    0:    0:    0:    0:    0:    0:    0:    0
>100_SECS: 0
HIST_TOTAL:      46770
```

Section 9. Changes and Fixes in this Release of Netperf

Revision 2.1 of netperf contains the following changes and fixes:

- Netperf now supports revision 3.51 of the Windows NT operating system (TCP and UDP socket tests only). The same code/binaries may actually run under Windows95, but that is untested as of this writing.
- The XTI test suite is now complete and supported (at least on HP−UX). This is for XTI Version 2. (−DDO_XTI compilation only)
- Fixes that can affect −DUSE_LOOPER CPU utilization.
- Support for TCP and UDP over IPv6 (IPng) using the BSD sockets interface (−DDO_IPV6 compilation only)
- Compiler support for a "long long" data type is only required for −DUSE_PSTAT compilation on HP−UX.
- The annotated examples in the manual are no longer implemented with screen dumps, making the manual *much* smaller.

Section 10. Known Bugs and Misfeatures

Revision 2.1 of netperf contains the following known bugs and misfeatures:

- Verbose Output – the logic for greater than standard verbosity is flawed in all tests except the TCP_STREAM test. The verbose output for the TCP_STREAM test could use some formatting help. When confidence intervals are requested, the verbose output will likely be flawed.
- On some systems, the UDP_RR test will fail with with the message "send_udp_rr: data recv error: Connection Refused." It is unknown if this is a bug in those systems for connected UDP sockets, or a bug in netperf.
- Some test suites support for neither histograms nor confidence intervals.
- The benchmark is not written to ANSI C. However, an ANSI C compiler is required for –DHISTOGRAM compilation. Future versions of the benchmark may require ANSI C for all modes of compilation. –DUSE_PSTAT compilation on HP–UX requires support for the "long long" data type.
- The manual does not show examples for tests other than UDP or TCP.
- The DLPI Tests are not fully debugged for multivendor or non–Ethernet environments.
- The Fore API Tests need better headers.
- The errors reported for remote netperf errors in a Fore API test will almost certainly be wrong as netperf does not distinguish between an atm_errno and a standard errno.
- The manual needs additional troubleshooting examples.
- CPU utilization measurements in Win32 (–DUSE_LOOPER) need calibration.

If you are feeling adventurous, fixes for these problems would be greatly appreciated ;–)

Section 11. Troubleshooting

Netperf is a rather simple tool, so its error conditions will be relatively simple and few in number. This section will cover some of the more common error messages and situations you might see and offers advice as to what might be wrong and how to correct it. The error messages displayed on non−HP−UX systems might not be the same.

establish_control: control socket connect failed: Connection refused

When this message is displayed, it can mean one of the following:

- Netperf has not been installed on the remote system, or you forgot to run the netserver program explicitly. Either install netperf following the instructions at the beginning of this document, or run netserver explicitly on the remote system.
- You made a typo in the files */etc/services*, or */etc/inetd.conf*. Check your entries against those in the installation section. If changes are needed, make them and then reconfigure inetd as described in the installation section.
- You forgot to reconfigure inetd. Reconfigure inetd following the instructions in the installation section.
- You specified a port number with the −p option to netperf that is not the same as the port number being used by the netserver program on the remote system.
- Inetd security on the remote system is not configured to allow connections from the local system. Follow the instructions in the manpage for inetd.sec.
- Netperf hates you. It has always hated you. Consult the job listings in misc.jobs.offered and start your life over :)

udp_send: data send error: Message too long
−or−
send_udp_rr: data send error: Message too long

These messages indicate the following:

- You have requested a send size (−m option) or request size (−r option) that is larger than the local socket send buffer size (−s option) with a UDP_STREAM or UDP_RR test (−t option). You should either increase the size of the socket buffer, or decrease the size of the send/request.

put_control: acknowledgement error wanted 6 got 5
netperf: dl_open: could not sent control message, errno = 0
netperf: send_dlpi_co_stream: dlpi_stream data descriptor: Error 0

This stream of messages indicates the following:

- You have specified a DLPI PPA that is not valid. Verify the correct PPA and try again. Variations on this stream of messages can be seen for each of the DLPI tests.

netperf: dl_open: open of /dev/foo failed, errno = 2
netperf: send_dlpi_co_stream: dlpi stream data descriptor: No such file or directory

This stream of messages indicates the following:

- You have specified a DLPI device file that is not valid. Verify the device file name and try again. Defaults for DLPI device files will vary from OS to OS. Variations on this stream of messages can be seen for each of the DLPI tests.

netperf: receive_response: no response received

This message indicates that netperf was expecting to receive a response on its control connection, but none was received within the timeout period. In some cases, this could be the result of the operating system restarting system calls that netperf/netserver expects to exit with an errno of EINTR.

Of course, there may be other problems that are not covered by this section. For further support, you can direct email questions to the author:

raj@cup.hp.com

There is also an Internet mailing list devoted to the discussion of netperf. It is called netperf–talk and is hosted on netperf.cup.hp.com. To subscribe, send an email message to netperf–talk–request@netperf.cup.hp.com. Please do *not* send subscription requests to netperf–talk itself.

Alternatively, the authors of netperf do try to read the Internet newsgroups comp.sys.hp.*, comp.protocols.tcp–ip, comp.benchmarks and the HP internal newsgroups pertaining to networking and performance. As likely as not, they will also see posts about netperf in other groups as well.

Section 12. Glossary

DLPI	**D**ata **L**ink **P**rovider **I**nterface. This is a standardized Link–Level Access (Level 2 of the OSI Reference Model) API. This interface is discussed in many books on Streams programming. The DLPI Version 2 Reference can be retrieved via anonymous FTP from col.hp.com.
TCP	**T**ransmission **C**ontrol **P**rotocol. This defines a reliable, byte–stream protocol for exchanging information between two hosts.
UDP	**U**ser **D**atagram **P**rotocol. This defines an unreliable, message–oriented protocol for exchanging information between two hosts.
TLI	**T**ransport **L**ayer **I**nterface. This is a standardized Transport level (Level 4 of the OSI Reference Model) API. This interface can be used in conjunction with many different transport protocols, including TCP, and the OSI Transports (TP0 through TP4). Often found in Streams implementations.
XTI	X/Open Transport Interface. This is a TLI–like (very TLI–like actually) Transport level API. The definition of XTI is maintained by X/Open.
IP	**I**nternet **P**rotocol. This protocol is the "glue" between TCP/UDP and the Link–Level. It provides the services of routing and packet segmentation and reassembly to export the illusion of a single homogenous network to the higher protocols.

Section 13. The Netperf Database

An online database of netperf results is available to anyone with a forms−capable WWW browser and access to the Internet. The URL to follow is:

http://www.cup.hp.com/netperf/NetperfPage.html

From there you can search or browse the database, or better still, submit numbers! There are additional links which will allow you to download copies of the benchmark or find other sources of network performance information.

Section 14. Netperf Futures

The netperf benchmark is becoming very large. In fact, it is almost large enough to be untenable – it certainly appears daunting to anyone wishing to port it to new platforms. So, unless overwhelming response is received in support of the following test suites, revision 2.1 will be the last release of netperf with the following tests:

- Unix Domain Sockets
- HP HiPPI LLA

Further, the following are also mildly under consideration for removal in the future:

- Fore ATM API

A good way to show overwhelming support for any of the test suites listed above would be to submit results using those suites to the netperf results database.

Appendix A. Netperf on NT

Netperf is not run quite the same on NT as it is under Unix. These differences stem from a (perceived) lack of certain services of Unix − namely the inetd. The lack of inetd support means that netserver must be run explicitly for each invocation of netperf. One simple mechanism to save time is to use the "for" functionality of the MS−DOS command shell:

$ **for %i in (1 2 3 ... N) do netserver −p 12865**

which will run N netservers in succession, one after the other. You can then run N successive instances of netperf on other systems before having to re−run netserver manually. If you are using scripts, you may wish to introduce some delays between individual invocations of netperf. Experimentation has shown that NT may not be able to get the next netserver running before the next netperf comes along. A pause of two seconds or so should be sufficient.

If you wish to run concurrent instances of netperf, it is necessary to combine the "for" and "start" commands of the MS−DOS command shell. On the system running netserver you would enter;

$ **for %i in (12865 12866 ... 12865+N) do start /B netserver −p %i**

which will run N parallel instances of the netserver, each waiting at their own port number. The port numbers you use are up to you, but they must match those you use in the invocation of netperf. For example, if you also wanted to run N parallel netperfs (perhaps to measure aggregate loopback performance) you would execute the following in another MS−DOS command shell:

$ **for %i in (12865 12866 ... 12865+N) do start /B netperf −p %i −P 0 −v 0 −l 240**

and then manually tabulate the results − after being certain that all tests finished closely to one another.

None of the scripts provided with netperf have been ported to MS−DOS command files for the 2.1 release. Any assistance you might be able to provide in that area would be greatly appreciated by the authors.

NAME

rlog − print log messages and other information about RCS files

SYNOPSIS

rlog [*options*] *file* . . .

DESCRIPTION

rlog prints information about RCS files.

Filenames matching an RCS suffix denote RCS files; all others denote working files. Names are paired as explained in **ci**(1).

rlog prints the following information for each RCS file: RCS file name, working file name, head (i.e., the number of the latest revision on the trunk), default branch, access list, locks, symbolic names, suffix, total number of revisions, number of revisions selected for printing, and descriptive text. This is followed by entries for the selected revisions in reverse chronological order for each branch. For each revision, **rlog** prints revision number, author, date/time, state, number of lines added/deleted (with respect to the previous revision), locker of the revision (if any), and log message. All times are displayed in Coordinated Universal Time (UTC) by default; this can be overridden with **−z**. Without options, **rlog** prints complete information. The options below restrict this output.

−L Ignore RCS files that have no locks set. This is convenient in combination with **−h**, **−l**, and **−R**.

−R Print only the name of the RCS file. This is convenient for translating a working file name into an RCS file name.

−h Print only the RCS file name, working file name, head, default branch, access list, locks, symbolic names, and suffix.

−t Print the same as **−h**, plus the descriptive text.

−N Do not print the symbolic names.

−b Print information about the revisions on the default branch, normally the highest branch on the trunk.

−d*dates*
Print information about revisions with a checkin date/time in the ranges given by the semicolon-separated list of *dates*. A range of the form *d1*<*d2* or *d2*>*d1* selects the revisions that were deposited between *d1* and *d2* exclusive. A range of the form <*d* or *d*> selects all revisions earlier than *d*. A range of the form *d*< or >*d* selects all revisions dated later than *d*. If < or > is followed by = then the ranges are inclusive, not exclusive. A range of the form *d* selects the single, latest revision dated *d* or earlier. The date/time strings *d*, *d1*, and *d2* are in the free format explained in **co**(1). Quoting is normally necessary, especially for < and >. Note that the separator is a semicolon.

−l[*lockers*]
Print information about locked revisions only. In addition, if the comma-separated list *lockers* of login names is given, ignore all locks other than those held by the *lockers*. For example, **rlog −L −R −lwft RCS/*** prints the name of RCS files locked by the user **wft**.

−r[*revisions*]
prints information about revisions given in the comma-separated list *revisions* of revisions and ranges. A range *rev1***:***rev2* means revisions *rev1* to *rev2* on the same branch, **:***rev* means revisions from the beginning of the branch up to and including *rev*, and *rev***:** means revisions starting with *rev* to the end of the branch containing *rev*. An argument that is a branch means all revisions on that branch. A range of branches means all revisions on the branches in that range. A branch followed by a **.** means the latest revision in that branch. A bare **−r** with no *revisions* means the latest revision on the default branch, normally the trunk.

−s*states*
prints information about revisions whose state attributes match one of the states given in the comma-separated list *states*.

−w[*logins*]
prints information about revisions checked in by users with login names appearing in the comma-separated list *logins*. If *logins* is omitted, the user's login is assumed.

−q This option has no effect; it is provided for consistency with other commands.

−T This option has no effect; it is present for compatibility with other RCS commands.

−V Print RCS's version number.

−V*n*
Emulate RCS version *n* when generating logs. See **co**(1) for more.

−x*suffixes*
Use *suffixes* to characterize RCS files. See **ci**(1) for details.

rlog prints the intersection of the revisions selected with the options **−d**, **−l**, **−s**, and **−w**, intersected with the union of the revisions selected by **−b** and **−r**.

−z*zone* specifies the date output format, and specifies the default time zone for *date* in the **−d***dates* option. The *zone* should be empty, a numeric UTC offset, or the special string **LT** for local time. The default is an empty *zone*, which uses the traditional RCS format of UTC without any time zone indication and with slashes separating the parts of the date; otherwise, times are output in ISO 8601 format with time zone indication. For example, if local time is January 11, 1990, 8pm Pacific Standard Time, eight hours west of UTC, then the time is output as follows:

option	*time output*	
−z	**1990/01/12 04:00:00**	*(default)*
−zLT	**1990-01-11 20:00:00−08**	
−z+05:30	**1990-01-12 09:30:00+05:30**	

EXAMPLES

```
rlog −L −R RCS/*
rlog −L −h RCS/*
rlog −L −l RCS/*
rlog RCS/*
```

The first command prints the names of all RCS files in the subdirectory **RCS** that have locks. The second command prints the headers of those files, and the third prints the headers plus the log messages of the locked revisions. The last command prints complete information.

ENVIRONMENT

RCSINIT
Options prepended to the argument list, separated by spaces. A backslash escapes spaces within an option. The **RCSINIT** options are prepended to the argument lists of most RCS commands. Useful **RCSINIT** options include **−q**, **−V**, **−x**, and **−z**.

RCS_MEM_LIMIT
An integer *lim*, measured in kilobytes, specifying the threshold under which commands will try to use memory-based operations for processing the RCS file. (For RCS files of size *lim* kilobytes or greater, RCS will use the slower standard input/output routines.) Default value is 256.

TMPDIR
Name of the temporary directory. If not set, the environment variables **TMP** and **TEMP** are inspected instead and the first value found is taken; if none of them are set, a host-dependent default is used, typically **/tmp**.

DIAGNOSTICS

The exit status is zero if and only if all operations were successful.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.8; Release Date: 2011-08-30.

SEE ALSO

ci(1), **co**(1), **ident**(1), **rcs**(1), **rcsdiff**(1), **rcsintro**(1), **rcsmerge**(1), **rcsfile**(5).

Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

BUGS

The separator for revision ranges in the **−r** option used to be − instead of **:**, but this leads to confusion when symbolic names contain −. For backwards compatibility **rlog −r** still supports the old − separator, but it warns about this obsolete use.

NAME

merge − three-way file merge

SYNOPSIS

merge [*options*] *file1 file2 file3*

DESCRIPTION

merge incorporates all changes that lead from *file2* to *file3* into *file1*. The result ordinarily goes into *file1*. **merge** is useful for combining separate changes to an original. Suppose *file2* is the original, and both *file1* and *file3* are modifications of *file2*. Then **merge** combines both changes.

A conflict occurs if both *file1* and *file3* have changes in a common segment of lines. If a conflict is found, **merge** normally outputs a warning and brackets the conflict with <<<<<<< and >>>>>>> lines. A typical conflict will look like this:

<<<<<<< file A
lines in file A
=======
lines in file B
>>>>>>> file B

If there are conflicts, the user should edit the result and delete one of the alternatives.

OPTIONS

−A Output conflicts using the **−A** style of **diff3**(1), if supported by **diff3**. This merges all changes leading from *file2* to *file3* into *file1*, and generates the most verbose output.

−E, **−e** These options specify conflict styles that generate less information than **−A**. See **diff3**(1) for details. The default is **−E**. With **−e**, **merge** does not warn about conflicts.

−L *label*
This option may be given up to three times, and specifies labels to be used in place of the corresponding file names in conflict reports. That is, **merge −L x −L y −L z a b c** generates output that looks like it came from files **x**, **y** and **z** instead of from files **a**, **b** and **c**.

−p Send results to standard output instead of overwriting *file1*.

−q Quiet; do not warn about conflicts.

−V Print RCS's version number.

DIAGNOSTICS

Exit status is 0 for no conflicts, 1 for some conflicts, 2 for trouble.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.8; Release Date: 2011-08-30.

SEE ALSO

diff3(1), **diff**(1), **rcsmerge**(1), **co**(1).

BUGS

It normally does not make sense to merge binary files as if they were text, but **merge** tries to do it anyway.

NAME

rcsintro − introduction to RCS commands

DESCRIPTION

The Revision Control System (RCS) manages multiple revisions of files. RCS automates the storing, retrieval, logging, identification, and merging of revisions. RCS is useful for text that is revised frequently, for example programs, documentation, graphics, papers, and form letters.

The basic user interface is extremely simple. The novice only needs to learn two commands: **ci**(1) and **co**(1). **ci**, short for "check in", deposits the contents of a file into an archival file called an RCS file. An RCS file contains all revisions of a particular file. **co**, short for "check out", retrieves revisions from an RCS file.

Functions of RCS

- Store and retrieve multiple revisions of text. RCS saves all old revisions in a space efficient way. Changes no longer destroy the original, because the previous revisions remain accessible. Revisions can be retrieved according to ranges of revision numbers, symbolic names, dates, authors, and states.
- Maintain a complete history of changes. RCS logs all changes automatically. Besides the text of each revision, RCS stores the author, the date and time of check-in, and a log message summarizing the change. The logging makes it easy to find out what happened to a module, without having to compare source listings or having to track down colleagues.
- Resolve access conflicts. When two or more programmers wish to modify the same revision, RCS alerts the programmers and prevents one modification from corrupting the other.
- Maintain a tree of revisions. RCS can maintain separate lines of development for each module. It stores a tree structure that represents the ancestral relationships among revisions.
- Merge revisions and resolve conflicts. Two separate lines of development of a module can be coalesced by merging. If the revisions to be merged affect the same sections of code, RCS alerts the user about the overlapping changes.
- Control releases and configurations. Revisions can be assigned symbolic names and marked as released, stable, experimental, etc. With these facilities, configurations of modules can be described simply and directly.
- Automatically identify each revision with name, revision number, creation time, author, etc. The identification is like a stamp that can be embedded at an appropriate place in the text of a revision. The identification makes it simple to determine which revisions of which modules make up a given configuration.
- Minimize secondary storage. RCS needs little extra space for the revisions (only the differences). If intermediate revisions are deleted, the corresponding deltas are compressed accordingly.

Getting Started with RCS

Suppose you have a file **f.c** that you wish to put under control of RCS. If you have not already done so, make an RCS directory with the command

mkdir RCS

Then invoke the check-in command

ci f.c

This command creates an RCS file in the **RCS** directory, stores **f.c** into it as revision 1.1, and deletes **f.c**. It also asks you for a description. The description should be a synopsis of the contents of the file. All later check-in commands will ask you for a log entry, which should summarize the changes that you made.

Files in the RCS directory are called RCS files; the others are called working files. To get back the working file **f.c** in the previous example, use the check-out command

co f.c

This command extracts the latest revision from the RCS file and writes it into **f.c**. If you want to edit **f.c**, you must lock it as you check it out with the command

co −l f.c

You can now edit **f.c**.

Suppose after some editing you want to know what changes that you have made. The command

rcsdiff f.c

tells you the difference between the most recently checked-in version and the working file. You can check the file back in by invoking

ci f.c

This increments the revision number properly.

If **ci** complains with the message

ci error: no lock set by *your name*

then you have tried to check in a file even though you did not lock it when you checked it out. Of course, it is too late now to do the check-out with locking, because another check-out would overwrite your modifications. Instead, invoke

rcs −l f.c

This command will lock the latest revision for you, unless somebody else got ahead of you already. In this case, you'll have to negotiate with that person.

Locking assures that you, and only you, can check in the next update, and avoids nasty problems if several people work on the same file. Even if a revision is locked, it can still be checked out for reading, compiling, etc. All that locking prevents is a *check-in* by anybody but the locker.

If your RCS file is private, i.e., if you are the only person who is going to deposit revisions into it, strict locking is not needed and you can turn it off. If strict locking is turned off, the owner of the RCS file need not have a lock for check-in; all others still do. Turning strict locking off and on is done with the commands

rcs −U f.c and **rcs −L f.c**

If you don't want to clutter your working directory with RCS files, create a subdirectory called **RCS** in your working directory, and move all your RCS files there. RCS commands will look first into that directory to find needed files. All the commands discussed above will still work, without any modification. (Actually, pairs of RCS and working files can be specified in three ways: (a) both are given, (b) only the working file is given, (c) only the RCS file is given. Both RCS and working files may have arbitrary names; RCS commands pair them up intelligently.)

To avoid the deletion of the working file during check-in (in case you want to continue editing or compiling), invoke

ci −l f.c or **ci −u f.c**

These commands check in **f.c** as usual, but perform an implicit check-out. The first form also locks the checked in revision, the second one doesn't. Thus, these options save you one check-out operation. The first form is useful if you want to continue editing, the second one if you just want to read the file. Both update the identification markers in your working file (see below).

You can give **ci** the number you want assigned to a checked in revision. Assume all your revisions were numbered 1.1, 1.2, 1.3, etc., and you would like to start release 2. The command

ci −r2 f.c or **ci −r2.1 f.c**

assigns the number 2.1 to the new revision. From then on, **ci** will number the subsequent revisions with 2.2, 2.3, etc. The corresponding **co** commands

co −r2 f.c and **co −r2.1 f.c**

retrieve the latest revision numbered 2.*x* and the revision 2.1, respectively. **co** without a revision number selects the latest revision on the *trunk*, i.e. the highest revision with a number consisting of two fields. Numbers with more than two fields are needed for branches. For example, to start a branch at revision 1.3, invoke

ci −r1.3.1 f.c

This command starts a branch numbered 1 at revision 1.3, and assigns the number 1.3.1.1 to the new revision. For more information about branches, see **rcsfile**(5).

Automatic Identification

RCS can put special strings for identification into your source and object code. To obtain such identification, place the marker

Id

into your text, for instance inside a comment. RCS will replace this marker with a string of the form

$Id: *file-name revision date time author state* **$**

With such a marker on the first page of each module, you can always see with which revision you are working. RCS keeps the markers up to date automatically. To propagate the markers into your object code, simply put them into literal character strings. In C, this is done as follows:

```
static char rcsid[] = "$Id$";
```

The command **ident** extracts such markers from any file, even object code and dumps. Thus, **ident** lets you find out which revisions of which modules were used in a given program.

You may also find it useful to put the marker **Log** into your text, inside a comment. This marker accumulates the log messages that are requested during check-in. Thus, you can maintain the complete history of your file directly inside it. There are several additional identification markers; see **co**(1) for details.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.8; Release Date: 2011-08-30.

SEE ALSO

ci(1), **co**(1), **ident**(1), **rcs**(1), **rcsdiff**(1), **rcsintro**(1), **rcsmerge**(1), **rlog**(1).

Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

NAME

rcsclean − clean up working files

SYNOPSIS

rcsclean [*options*] [*file* . . .]

DESCRIPTION

rcsclean removes files that are not being worked on. **rcsclean −u** also unlocks and removes files that are being worked on but have not changed.

For each *file* given, **rcsclean** compares the working file and a revision in the corresponding RCS file. If it finds a difference, it does nothing. Otherwise, it first unlocks the revision if the **−u** option is given, and then removes the working file unless the working file is writable and the revision is locked. It logs its actions by outputting the corresponding **rcs −u** and **rm −f** commands on the standard output.

Files are paired as explained in **ci**(1). If no *file* is given, all working files in the current directory are cleaned. Filenames matching an RCS suffix denote RCS files; all others denote working files.

The number of the revision to which the working file is compared may be attached to any of the options **−n**, **−q**, **−r**, or **−u**. If no revision number is specified, then if the **−u** option is given and the caller has one revision locked, **rcsclean** uses that revision; otherwise **rcsclean** uses the latest revision on the default branch, normally the root.

rcsclean is useful for **clean** targets in makefiles. See also **rcsdiff**(1), which prints out the differences, and **ci**(1), which normally reverts to the previous revision if a file was not changed.

OPTIONS

−k*subst*
: Use *subst* style keyword substitution when retrieving the revision for comparison. See **co**(1) for details.

−n[*rev*] Do not actually remove any files or unlock any revisions. Using this option will tell you what **rcsclean** would do without actually doing it.

−q[*rev*] Do not log the actions taken on standard output.

−r[*rev*] This option has no effect other than specifying the revision for comparison.

−T Preserve the modification time on the RCS file even if the RCS file changes because a lock is removed. This option can suppress extensive recompilation caused by a **make**(1) dependency of some other copy of the working file on the RCS file. Use this option with care; it can suppress recompilation even when it is needed, i.e. when the lock removal would mean a change to keyword strings in the other working file.

−u[*rev*] Unlock the revision if it is locked and no difference is found.

−V Print RCS's version number.

−V*n* Emulate RCS version *n*. See **co**(1) for details.

−x*suffixes*
: Use *suffixes* to characterize RCS files. See **ci**(1) for details.

−z*zone* Use *zone* as the time zone for keyword substitution; see **co**(1) for details.

EXAMPLES

rcsclean *.c *.h

removes all working files ending in **.c** or **.h** that were not changed since their checkout.

rcsclean

removes all working files in the current directory that were not changed since their checkout.

FILES

rcsclean accesses files much as **ci**(1) does.

ENVIRONMENT

RCSINIT

Options prepended to the argument list, separated by spaces. A backslash escapes spaces within an option. The **RCSINIT** options are prepended to the argument lists of most RCS commands. Useful **RCSINIT** options include **−q**, **−V**, **−x**, and **−z**.

RCS_MEM_LIMIT

An integer *lim*, measured in kilobytes, specifying the threshold under which commands will try to use memory-based operations for processing the RCS file. (For RCS files of size *lim* kilobytes or greater, RCS will use the slower standard input/output routines.) Default value is 256.

TMPDIR

Name of the temporary directory. If not set, the environment variables **TMP** and **TEMP** are inspected instead and the first value found is taken; if none of them are set, a host-dependent default is used, typically **/tmp**.

DIAGNOSTICS

The exit status is zero if and only if all operations were successful. Missing working files and RCS files are silently ignored.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.8; Release Date: 2011-08-30.

SEE ALSO

ci(1), **co**(1), **ident**(1), **rcs**(1), **rcsdiff**(1), **rcsintro**(1), **rcsmerge**(1), **rlog**(1), **rcsfile**(5).

Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

BUGS

At least one *file* must be given in older Unix versions that do not provide the needed directory scanning operations.

NAME

ident − identify RCS keyword strings in files

SYNOPSIS

ident [**−q**] [**−V**] [*file* . . .]

DESCRIPTION

ident searches for all instances of the pattern **$***keyword***:** *text* **$** in the named files or, if no files are named, the standard input.

These patterns are normally inserted automatically by the RCS command **co**(1), but can also be inserted manually. The option **−q** suppresses the warning given if there are no patterns in a file. The option **−V** prints RCS's version number.

ident works on text files as well as object files and dumps. For example, if the C program in **f.c** contains

```
#include <stdio.h>
static char const rcsid[] =
  "$Id: f.c,v 5.4 1993/11/09 17:40:15 eggert Exp $";
int main() { return printf("%s\n", rcsid) == EOF; }
```

and **f.c** is compiled into **f.o**, then the command

```
ident  f.c  f.o
```

will output

```
f.c:
   $Id: f.c,v 5.4 1993/11/09 17:40:15 eggert Exp $
f.o:
   $Id: f.c,v 5.4 1993/11/09 17:40:15 eggert Exp $
```

If a C program defines a string like **rcsid** above but does not use it, **lint**(1) may complain, and some C compilers will optimize away the string. The most reliable solution is to have the program use the **rcsid** string, as shown in the example above.

ident finds all instances of the **$***keyword***:** *text* **$** pattern, even if *keyword* is not actually an RCS-supported keyword. This gives you information about nonstandard keywords like **$XConsortium$**.

KEYWORDS

Here is the list of keywords currently maintained by **co**(1). All times are given in Coordinated Universal Time (UTC, sometimes called GMT) by default, but if the files were checked out with **co**'s **−z***zone* option, times are given with a numeric time zone indication appended.

$Author$
The login name of the user who checked in the revision.

$Date$ The date and time the revision was checked in.

$Header$
A standard header containing the full RCS file name, the revision number, the date and time, the author, the state, and the locker (if locked).

Id Same as **$Header$**, except that the RCS file name is without directory components.

$Locker$
The login name of the user who locked the revision (empty if not locked).

Log The log message supplied during checkin. For **ident**'s purposes, this is equivalent to **$RCSfile$**.

$Name$
The symbolic name used to check out the revision, if any.

$RCSfile$
The RCS file name without directory components.

$Revision$
The revision number assigned to the revision.

$Source$
The full RCS file name.

$State$
The state assigned to the revision with the **−s** option of **rcs**(1) or **ci**(1).

co(1) represents the following characters in keyword values by escape sequences to keep keyword strings well-formed.

char	*escape sequence*
tab	**\t**
newline	**\n**
space	**\040**
$	**\044**
\	\\

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.8; Release Date: 2011-08-30.

SEE ALSO

ci(1), **co**(1), **rcs**(1), **rcsdiff**(1), **rcsintro**(1), **rcsmerge**(1), **rlog**(1), **rcsfile**(5).

Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

NAME

co − check out RCS revisions

SYNOPSIS

co [*options*] *file* . . .

DESCRIPTION

co retrieves a revision from each RCS file and stores it into the corresponding working file.

Filenames matching an RCS suffix denote RCS files; all others denote working files. Names are paired as explained in **ci**(1).

Revisions of an RCS file can be checked out locked or unlocked. Locking a revision prevents overlapping updates. A revision checked out for reading or processing (e.g., compiling) need not be locked. A revision checked out for editing and later checkin must normally be locked. Checkout with locking fails if the revision to be checked out is currently locked by another user. (A lock can be broken with **rcs**(1).) Checkout with locking also requires the caller to be on the access list of the RCS file, unless he is the owner of the file or the superuser, or the access list is empty. Checkout without locking is not subject to accesslist restrictions, and is not affected by the presence of locks.

A revision is selected by options for revision or branch number, checkin date/time, author, or state. When the selection options are applied in combination, **co** retrieves the latest revision that satisfies all of them. If none of the selection options is specified, **co** retrieves the latest revision on the default branch (normally the trunk, see the **−b** option of **rcs**(1)). A revision or branch number can be attached to any of the options **−f**, **−I**, **−l**, **−M**, **−p**, **−q**, **−r**, or **−u**. The options **−d** (date), **−s** (state), and **−w** (author) retrieve from a single branch, the *selected* branch, which is either specified by one of **−f**, . . ., **−u**, or the default branch.

A **co** command applied to an RCS file with no revisions creates a zero-length working file. **co** always performs keyword substitution (see below).

OPTIONS

−r[*rev*] retrieves the latest revision whose number is less than or equal to *rev*. If *rev* indicates a branch rather than a revision, the latest revision on that branch is retrieved. If *rev* is omitted, the latest revision on the default branch (see the **−b** option of **rcs**(1)) is retrieved. If *rev* is **$**, **co** determines the revision number from keyword values in the working file. Otherwise, a revision is composed of one or more numeric or symbolic fields separated by periods. If *rev* begins with a period, then the default branch (normally the trunk) is prepended to it. If *rev* is a branch number followed by a period, then the latest revision on that branch is used. The numeric equivalent of a symbolic field is specified with the **−n** option of the commands **ci**(1) and **rcs**(1).

−l[*rev*] same as **−r**, except that it also locks the retrieved revision for the caller.

−u[*rev*] same as **−r**, except that it unlocks the retrieved revision if it was locked by the caller. If *rev* is omitted, **−u** retrieves the revision locked by the caller, if there is one; otherwise, it retrieves the latest revision on the default branch.

−f[*rev*] forces the overwriting of the working file; useful in connection with **−q**. See also FILE MODES below.

−kkv Generate keyword strings using the default form, e.g. **$Revision: 5.8 $** for the **Revision** keyword. A locker's name is inserted in the value of the **Header**, **Id**, and **Locker** keyword strings only as a file is being locked, i.e. by **ci −l** and **co −l**. This is the default.

−kkvl Like **−kkv**, except that a locker's name is always inserted if the given revision is currently locked.

−kk Generate only keyword names in keyword strings; omit their values. See KEYWORD SUBSTITUTION below. For example, for the **Revision** keyword, generate the string **$Revision$** instead of **$Revision: 5.8 $**. This option is useful to ignore differences due to keyword substitution when comparing different revisions of a file. Log messages are inserted after **Log** keywords even if **−kk** is specified, since this tends to be more useful when merging changes.

−ko Generate the old keyword string, present in the working file just before it was checked in. For example, for the **Revision** keyword, generate the string **$Revision: 1.1 $** instead of **$Revision: 5.8**

$ if that is how the string appeared when the file was checked in. This can be useful for file formats that cannot tolerate any changes to substrings that happen to take the form of keyword strings.

−kb Generate a binary image of the old keyword string. This acts like **−ko**, except it performs all working file input and output in binary mode. This makes little difference on Posix and Unix hosts, but on DOS-like hosts one should use **rcs −i −kb** to initialize an RCS file intended to be used for binary files. Also, on all hosts, **rcsmerge**(1) normally refuses to merge files when **−kb** is in effect.

−kv Generate only keyword values for keyword strings. For example, for the **Revision** keyword, generate the string **5.8** instead of **$Revision: 5.8 $**. This can help generate files in programming languages where it is hard to strip keyword delimiters like **$Revision: $** from a string. However, further keyword substitution cannot be performed once the keyword names are removed, so this option should be used with care. Because of this danger of losing keywords, this option cannot be combined with **−l**, and the owner write permission of the working file is turned off; to edit the file later, check it out again without **−kv**.

−p[*rev*] prints the retrieved revision on the standard output rather than storing it in the working file. This option is useful when **co** is part of a pipe.

−q[*rev*] quiet mode; diagnostics are not printed.

−I[*rev*] interactive mode; the user is prompted and questioned even if the standard input is not a terminal.

−d*date* retrieves the latest revision on the selected branch whose checkin date/time is less than or equal to *date*. The date and time can be given in free format. The time zone **LT** stands for local time; other common time zone names are understood. For example, the following *date*s are equivalent if local time is January 11, 1990, 8pm Pacific Standard Time, eight hours west of Coordinated Universal Time (UTC):

8:00 pm lt	
4:00 AM, Jan. 12, 1990	default is UTC
1990-01-12 04:00:00+00	ISO 8601 (UTC)
1990-01-11 20:00:00−08	ISO 8601 (local time)
1990/01/12 04:00:00	traditional RCS format
Thu Jan 11 20:00:00 1990 LT	output of **ctime**(3) + **LT**
Thu Jan 11 20:00:00 PST 1990	output of **date**(1)
Fri Jan 12 04:00:00 GMT 1990	
Thu, 11 Jan 1990 20:00:00 −0800	Internet RFC 822
12-January-1990, 04:00 WET	

Most fields in the date and time can be defaulted. The default time zone is normally UTC, but this can be overridden by the **−z** option. The other defaults are determined in the order year, month, day, hour, minute, and second (most to least significant). At least one of these fields must be provided. For omitted fields that are of higher significance than the highest provided field, the time zone's current values are assumed. For all other omitted fields, the lowest possible values are assumed. For example, without **−z**, the date **20, 10:30** defaults to 10:30:00 UTC of the 20th of the UTC time zone's current month and year. The date/time must be quoted if it contains spaces.

−M[*rev*]
Set the modification time on the new working file to be the date of the retrieved revision. Use this option with care; it can confuse **make**(1).

−s*state* retrieves the latest revision on the selected branch whose state is set to *state*.

−T Preserve the modification time on the RCS file even if the RCS file changes because a lock is added or removed. This option can suppress extensive recompilation caused by a **make**(1) dependency of some other copy of the working file on the RCS file. Use this option with care; it can suppress recompilation even when it is needed, i.e. when the change of lock would mean a change to keyword strings in the other working file.

−w[*login*]
retrieves the latest revision on the selected branch which was checked in by the user with login name *login*. If the argument *login* is omitted, the caller's login is assumed.

−j *joinlist*
generates a new revision which is the join of the revisions on *joinlist*. This option is largely obsoleted by **rcsmerge**(1) but is retained for backwards compatibility.

The *joinlist* is a comma-separated list of pairs of the form *rev2***:***rev3*, where *rev2* and *rev3* are (symbolic or numeric) revision numbers. For the initial such pair, *rev1* denotes the revision selected by the above options **−f**, . . ., **−w**. For all other pairs, *rev1* denotes the revision generated by the previous pair. (Thus, the output of one join becomes the input to the next.)

For each pair, **co** joins revisions *rev1* and *rev3* with respect to *rev2*. This means that all changes that transform *rev2* into *rev1* are applied to a copy of *rev3*. This is particularly useful if *rev1* and *rev3* are the ends of two branches that have *rev2* as a common ancestor. If *rev1*<*rev2*<*rev3* on the same branch, joining generates a new revision which is like *rev3*, but with all changes that lead from *rev1* to *rev2* undone. If changes from *rev2* to *rev1* overlap with changes from *rev2* to *rev3*, **co** reports overlaps as described in **merge**(1).

For the initial pair, *rev2* can be omitted. The default is the common ancestor. If any of the arguments indicate branches, the latest revisions on those branches are assumed. The options **−l** and **−u** lock or unlock *rev1*.

−V Print RCS's version number.

−V*n* Emulate RCS version *n*, where *n* can be **3**, **4**, or **5**. This can be useful when interchanging RCS files with others who are running older versions of RCS. To see which version of RCS your correspondents are running, have them invoke **rcs −V**; this works with newer versions of RCS. If it doesn't work, have them invoke **rlog** on an RCS file; if none of the first few lines of output contain the string **branch:** it is version 3; if the dates' years have just two digits, it is version 4; otherwise, it is version 5. An RCS file generated while emulating version 3 loses its default branch. An RCS revision generated while emulating version 4 or earlier has a time stamp that is off by up to 13 hours. A revision extracted while emulating version 4 or earlier contains abbreviated dates of the form *yy*/*mm*/*dd* and can also contain different white space and line prefixes in the substitution for **Log**.

−x*suffixes*
Use *suffixes* to characterize RCS files. See **ci**(1) for details.

−z*zone* specifies the date output format in keyword substitution, and specifies the default time zone for *date* in the **−d***date* option. The *zone* should be empty, a numeric UTC offset, or the special string **LT** for local time. The default is an empty *zone*, which uses the traditional RCS format of UTC without any time zone indication and with slashes separating the parts of the date; otherwise, times are output in ISO 8601 format with time zone indication. For example, if local time is January 11, 1990, 8pm Pacific Standard Time, eight hours west of UTC, then the time is output as follows:

option	*time output*	
−z	**1990/01/12 04:00:00**	*(default)*
−zLT	**1990-01-11 20:00:00−08**	
−z+05:30	**1990-01-12 09:30:00+05:30**	

The **−z** option does not affect dates stored in RCS files, which are always UTC.

KEYWORD SUBSTITUTION

Strings of the form **$***keyword***$** and **$***keyword***:**. . .**$** embedded in the text are replaced with strings of the form **$***keyword***:***value***$** where *keyword* and *value* are pairs listed below. Keywords can be embedded in literal strings or comments to identify a revision.

Initially, the user enters strings of the form **$***keyword***$**. On checkout, **co** replaces these strings with strings of the form **$***keyword***:***value***$**. If a revision containing strings of the latter form is checked back in, the value fields will be replaced during the next checkout. Thus, the keyword values are automatically updated

on checkout. This automatic substitution can be modified by the **−k** options.

Keywords and their corresponding values:

$Author$
The login name of the user who checked in the revision.

$Date$ The date and time the revision was checked in. With **−z***zone* a numeric time zone offset is appended; otherwise, the date is UTC.

$Header$
A standard header containing the full RCS file name, the revision number, the date and time, the author, the state, and the locker (if locked). With **−z***zone* a numeric time zone offset is appended to the date; otherwise, the date is UTC.

Id Same as **$Header$**, except that the RCS file name is without the directory components.

$Locker$
The login name of the user who locked the revision (empty if not locked).

Log The log message supplied during checkin, preceded by a header containing the RCS file name, the revision number, the author, and the date and time. With **−z***zone* a numeric time zone offset is appended; otherwise, the date is UTC. Existing log messages are *not* replaced. Instead, the new log message is inserted after **$Log:**...**$**. This is useful for accumulating a complete change log in a source file.

Each inserted line is prefixed by the string that prefixes the **Log** line. For example, if the **Log** line is "// **$Log: tan.cc $**", RCS prefixes each line of the log with "// ". This is useful for languages with comments that go to the end of the line. The convention for other languages is to use a " * " prefix inside a multiline comment. For example, the initial log comment of a C program conventionally is of the following form:

```
/*
 * $Log$
 */
```

For backwards compatibility with older versions of RCS, if the log prefix is /* or **(*** surrounded by optional white space, inserted log lines contain a space instead of / or **(**; however, this usage is obsolescent and should not be relied on.

$Name$
The symbolic name used to check out the revision, if any. For example, **co −rJoe** generates **$Name: Joe $**. Plain **co** generates just **$Name: $**.

$RCSfile$
The RCS file name without directory components.

$Revision$
The revision number assigned to the revision.

$Source$
The full RCS file name.

$State$
The state assigned to the revision with the **−s** option of **rcs**(1) or **ci**(1).

The following characters in keyword values are represented by escape sequences to keep keyword strings well-formed.

char	*escape sequence*
tab	**\t**
newline	**\n**
space	**\040**
$	**\044**
\	\\

FILE MODES

The working file inherits the read and execute permissions from the RCS file. In addition, the owner write permission is turned on, unless **−kv** is set or the file is checked out unlocked and locking is set to strict (see **rcs**(1)).

If a file with the name of the working file exists already and has write permission, **co** aborts the checkout, asking beforehand if possible. If the existing working file is not writable or **−f** is given, the working file is deleted without asking.

FILES

co accesses files much as **ci**(1) does, except that it does not need to read the working file unless a revision number of **$** is specified.

ENVIRONMENT

RCSINIT

Options prepended to the argument list, separated by spaces. A backslash escapes spaces within an option. The **RCSINIT** options are prepended to the argument lists of most RCS commands. Useful **RCSINIT** options include **−q**, **−V**, **−x**, and **−z**.

RCS_MEM_LIMIT

An integer *lim*, measured in kilobytes, specifying the threshold under which commands will try to use memory-based operations for processing the RCS file. (For RCS files of size *lim* kilobytes or greater, RCS will use the slower standard input/output routines.) Default value is 256.

TMPDIR

Name of the temporary directory. If not set, the environment variables **TMP** and **TEMP** are inspected instead and the first value found is taken; if none of them are set, a host-dependent default is used, typically **/tmp**.

DIAGNOSTICS

The RCS file name, the working file name, and the revision number retrieved are written to the diagnostic output. The exit status is zero if and only if all operations were successful.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.8; Release Date: 2011-08-30.

SEE ALSO

rcsintro(1), **ci**(1), **ctime**(3), **date**(1), **ident**(1), **make**(1), **rcs**(1), **rcsclean**(1), **rcsdiff**(1), **rcsmerge**(1), **rlog**(1), **rcsfile**(5).

Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

LIMITS

Links to the RCS and working files are not preserved.

There is no way to selectively suppress the expansion of keywords, except by writing them differently. In nroff and troff, this is done by embedding the null-character **\&** into the keyword.

NAME

rcsfile − RCS file format

DESCRIPTION

An RCS file's contents are described by the grammar below.

The text is free format: space, backspace, tab, newline, vertical tab, form feed, and carriage return (collectively, *white space*) have no significance except in strings. However, white space cannot appear within an id, num, or sym, and an RCS file must end with a newline.

Strings are enclosed by **@**. If a string contains a **@**, it must be doubled; otherwise, strings can contain arbitrary binary data.

The meta syntax uses the following conventions: '|' (bar) separates alternatives; '{' and '}' enclose optional phrases; '{' and '}*' enclose phrases that can be repeated zero or more times; '{' and '}+' enclose phrases that must appear at least once and can be repeated; Terminal symbols are in **boldface**; nonterminal symbols are in *italics*.

rcstext ::= *admin* {*delta*}* *desc* {*deltatext*}*

admin ::= **head** {*num*}**;**
{ **branch** {*num*}**;** }
access {*id*}***;**
symbols {*sym* **:** *num*}***;**
locks {*id* **:** *num*}***;** {**strict** **;**}
{ **integrity** {*string*}**;** }
{ **comment** {*string*}**;** }
{ **expand** {*string*}**;** }

delta ::= *num*
date *num***;**
author *id***;**
state {*id*}**;**
branches {*num*}***;**
next {*num*}**;**
{ **commitid** *sym***;** }

desc ::= **desc** *string*

deltatext ::= *num*
log *string*
text *string*

num ::= {*digit* | **.**}+

digit ::= **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

id ::= {*idchar* | **.**}+

sym ::= {*idchar*}+

idchar ::= any visible graphic character except *special*

special ::= **$** | **,** | **.** | **:** | **;** | **@**

string ::= **@**{any character, with **@** doubled}***@**

word ::= *id* | *num* | *string* | **:**

Identifiers are case sensitive. Keywords are in lower case only. The sets of keywords and identifiers can overlap. In most environments RCS uses the ISO 8859/1 encoding: visible graphic characters are codes 041−176 and 240−377, and white space characters are codes 010−015 and 040.

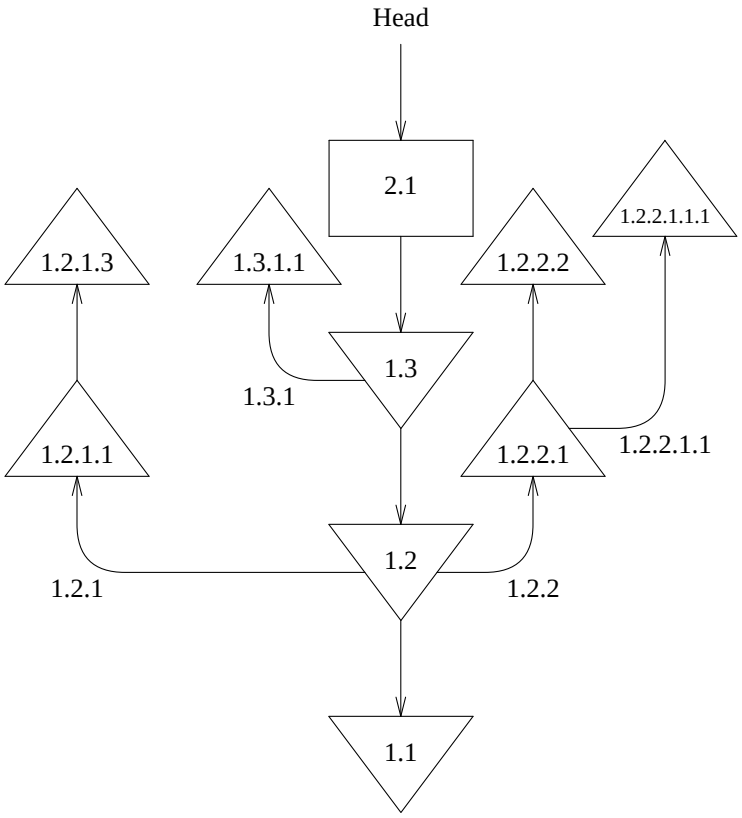
Dates, which appear after the **date** keyword, are of the form ***Y.mm.dd.hh.mm.ss***, where *Y* is the year, *mm* the month (01−12), *dd* the day (01−31), *hh* the hour (00−23), *mm* the minute (00−59), and *ss* the second (00−60). *Y* contains just the last two digits of the year for years from 1900 through 1999, and all the digits

of years thereafter. Dates use the Gregorian calendar; times use UTC.

The *delta* nodes form a tree. All nodes whose numbers consist of a single pair (e.g., 2.3, 2.1, 1.3, etc.) are on the trunk, and are linked through the **next** field in order of decreasing numbers. The **head** field in the *admin* node points to the head of that sequence (i.e., contains the highest pair). The **branch** node in the admin node indicates the default branch (or revision) for most RCS operations. If empty, the default branch is the highest branch on the trunk.

All *delta* nodes whose numbers consist of $2n$ fields ($n \geq 2$) (e.g., 3.1.1.1, 2.1.2.2, etc.) are linked as follows. All nodes whose first $2n-1$ number fields are identical are linked through the **next** field in order of increasing numbers. For each such sequence, the *delta* node whose number is identical to the first $2n-2$ number fields of the deltas on that sequence is called the branchpoint. The **branches** field of a node contains a list of the numbers of the first nodes of all sequences for which it is a branchpoint. This list is ordered in increasing numbers.

The following diagram shows an example of an RCS file's organization.



IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.8; Release Date: 2011-08-30.

SEE ALSO

rcsintro(1), **ci**(1), **co**(1), **ident**(1), **rcs**(1), **rcsclean**(1), **rcsdiff**(1), **rcsmerge**(1), **rlog**(1).

Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

NAME

rcsmerge − merge RCS revisions

SYNOPSIS

rcsmerge [*options*] *file*

DESCRIPTION

rcsmerge incorporates the changes between two revisions of an RCS file into the corresponding working file.

Filenames matching an RCS suffix denote RCS files; all others denote working files. Names are paired as explained in **ci**(1).

At least one revision must be specified with one of the options described below, usually **−r**. At most two revisions may be specified. If only one revision is specified, the latest revision on the default branch (normally the highest branch on the trunk) is assumed for the second revision. Revisions may be specified numerically or symbolically.

rcsmerge prints a warning if there are overlaps, and delimits the overlapping regions as explained in **merge**(1). The command is useful for incorporating changes into a checked-out revision.

OPTIONS

−A Output conflicts using the **−A** style of **diff3**(1), if supported by **diff3**. This merges all changes leading from *file2* to *file3* into *file1*, and generates the most verbose output.

−E, **−e** These options specify conflict styles that generate less information than **−A**. See **diff3**(1) for details. The default is **−E**. With **−e**, **rcsmerge** does not warn about conflicts.

−k*subst*
Use *subst* style keyword substitution. See **co**(1) for details. For example, **−kk −r1.1 −r1.2** ignores differences in keyword values when merging the changes from **1.1** to **1.2**. It normally does not make sense to merge binary files as if they were text, so **rcsmerge** refuses to merge files if **−kb** expansion is used.

−p[*rev*] Send the result to standard output instead of overwriting the working file.

−q[*rev*] Run quietly; do not print diagnostics.

−r[*rev*] Merge with respect to revision *rev*. Here an empty *rev* stands for the latest revision on the default branch, normally the head.

−T This option has no effect; it is present for compatibility with other RCS commands.

−V Print RCS's version number.

−V*n* Emulate RCS version *n*. See **co**(1) for details.

−x*suffixes*
Use *suffixes* to characterize RCS files. See **ci**(1) for details.

−z*zone* Use *zone* as the time zone for keyword substitution. See **co**(1) for details.

EXAMPLES

Suppose you have released revision 2.8 of **f.c**. Assume furthermore that after you complete an unreleased revision 3.4, you receive updates to release 2.8 from someone else. To combine the updates to 2.8 and your changes between 2.8 and 3.4, put the updates to 2.8 into file f.c and execute

```
rcsmerge −p −r2.8 −r3.4 f.c >f.merged.c
```

Then examine **f.merged.c**. Alternatively, if you want to save the updates to 2.8 in the RCS file, check them in as revision 2.8.1.1 and execute **co −j**:

```
ci −r2.8.1.1 f.c
co −r3.4 −j2.8:2.8.1.1 f.c
```

As another example, the following command undoes the changes between revision 2.4 and 2.8 in your currently checked out revision in **f.c**.

rcsmerge −r2.8 −r2.4 f.c

Note the order of the arguments, and that **f.c** will be overwritten.

ENVIRONMENT

RCSINIT

Options prepended to the argument list, separated by spaces. A backslash escapes spaces within an option. The **RCSINIT** options are prepended to the argument lists of most RCS commands. Useful **RCSINIT** options include **−q**, **−V**, **−x**, and **−z**.

RCS_MEM_LIMIT

An integer *lim*, measured in kilobytes, specifying the threshold under which commands will try to use memory-based operations for processing the RCS file. (For RCS files of size *lim* kilobytes or greater, RCS will use the slower standard input/output routines.) Default value is 256.

TMPDIR

Name of the temporary directory. If not set, the environment variables **TMP** and **TEMP** are inspected instead and the first value found is taken; if none of them are set, a host-dependent default is used, typically **/tmp**.

DIAGNOSTICS

Exit status is 0 for no overlaps, 1 for some overlaps, 2 for trouble.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.8; Release Date: 2011-08-30.

SEE ALSO

ci(1), **co**(1), **ident**(1), **merge**(1), **rcs**(1), **rcsdiff**(1), **rcsintro**(1), **rlog**(1), **rcsfile**(5).

Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

NAME

rcs − change RCS file attributes

SYNOPSIS

rcs *options file* ...

DESCRIPTION

rcs creates new RCS files or changes attributes of existing ones. An RCS file contains multiple revisions of text, an access list, a change log, descriptive text, and some control attributes. For **rcs** to work, the caller's login name must be on the access list, except if the access list is empty, the caller is the owner of the file or the superuser, or the **−i** option is present.

Filenames matching an RCS suffix denote RCS files; all others denote working files. Names are paired as explained in **ci**(1). Revision numbers use the syntax described in **ci**(1).

OPTIONS

−i Create and initialize a new RCS file, but do not deposit any revision. If the RCS file name has no directory component, try to place it first into the subdirectory **./RCS**, and then into the current directory. If the RCS file already exists, print an error message.

−a*logins*
Append the login names appearing in the comma-separated list *logins* to the access list of the RCS file.

−A*oldfile*
Append the access list of *oldfile* to the access list of the RCS file.

−e[*logins*]
Erase the login names appearing in the comma-separated list *logins* from the access list of the RCS file. If *logins* is omitted, erase the entire access list.

−b[*rev*] Set the default branch to *rev*. If *rev* is omitted, the default branch is reset to the (dynamically) highest branch on the trunk.

−c*string*
Set the comment leader to *string*. An initial **ci**, or an **rcs −i** without **−c**, guesses the comment leader from the suffix of the working file name.

This option is obsolescent, since RCS normally uses the preceding **Log** line's prefix when inserting log lines during checkout (see **co**(1)). However, older versions of RCS use the comment leader instead of the **Log** line's prefix, so if you plan to access a file with both old and new versions of RCS, make sure its comment leader matches its **Log** line prefix.

−k*subst*
Set the default keyword substitution to *subst*. The effect of keyword substitution is described in **co**(1). Giving an explicit **−k** option to **co**, **rcsdiff**, and **rcsmerge** overrides this default. Beware **rcs −kv**, because **−kv** is incompatible with **co −l**. Use **rcs −kkv** to restore the normal default keyword substitution.

−l[*rev*] Lock the revision with number *rev*. If a branch is given, lock the latest revision on that branch. If *rev* is omitted, lock the latest revision on the default branch. Locking prevents overlapping changes. If someone else already holds the lock, the lock is broken as with **rcs −u** (see below).

−u[*rev*] Unlock the revision with number *rev*. If a branch is given, unlock the latest revision on that branch. If *rev* is omitted, remove the latest lock held by the caller. Normally, only the locker of a revision can unlock it. Somebody else unlocking a revision breaks the lock. If RCS was configured **−−with-mailer**, then this causes a mail message to be sent to the original locker. The message contains a commentary solicited from the breaker. The commentary is terminated by end-of-file or by a line containing **.** by itself.

−L Set locking to *strict*. Strict locking means that the owner of an RCS file is not exempt from locking for checkin. This option should be used for files that are shared.

−U Set locking to non-strict. Non-strict locking means that the owner of a file need not lock a revision for checkin. This option should *not* be used for files that are shared. Whether default locking is strict is determined by your system administrator, but it is normally strict.

−m*rev***:**[*msg*]
Replace revision *rev*'s log message with *msg*. If *msg* is omitted, it defaults to "*** empty log message ***".

−M Do not send mail when breaking somebody else's lock. This option is not meant for casual use; it is meant for programs that warn users by other means, and invoke **rcs −u** only as a low-level lock-breaking operation.

−n*name*[**:**[*rev*]]
Associate the symbolic name *name* with the branch or revision *rev*. Delete the symbolic name if both **:** and *rev* are omitted; otherwise, print an error message if *name* is already associated with another number. If *rev* is symbolic, it is expanded before association. A *rev* consisting of a branch number followed by a **.** stands for the current latest revision in the branch. A **:** with an empty *rev* stands for the current latest revision on the default branch, normally the trunk. For example, **rcs −n***name***: RCS/*** associates *name* with the current latest revision of all the named RCS files; this contrasts with **rcs −n***name***:$ RCS/*** which associates *name* with the revision numbers extracted from keyword strings in the corresponding working files.

−N*name*[**:**[*rev*]]
Act like **−n**, except override any previous assignment of *name*.

−o*range*
deletes ("outdates") the revisions given by *range*. A range consisting of a single revision number means that revision. A range consisting of a branch number means the latest revision on that branch. A range of the form *rev1***:***rev2* means revisions *rev1* to *rev2* on the same branch, **:***rev* means from the beginning of the branch containing *rev* up to and including *rev*, and *rev***:** means from revision *rev* to the end of the branch containing *rev*. None of the outdated revisions can have branches or locks.

−q Run quietly; do not print diagnostics.

−I Run interactively, even if the standard input is not a terminal.

−s*state*[**:***rev*]
Set the state attribute of the revision *rev* to *state*. If *rev* is a branch number, assume the latest revision on that branch. If *rev* is omitted, assume the latest revision on the default branch. Any identifier is acceptable for *state*. A useful set of states is **Exp** (for experimental), **Stab** (for stable), and **Rel** (for released). By default, **ci**(1) sets the state of a revision to **Exp**.

−t[*file*] Write descriptive text from the contents of the named *file* into the RCS file, deleting the existing text. The *file* name cannot begin with **−**. If *file* is omitted, obtain the text from standard input, terminated by end-of-file or by a line containing **.** by itself. Prompt for the text if interaction is possible; see **−I**. With **−i**, descriptive text is obtained even if **−t** is not given.

−t−*string*
Write descriptive text from the *string* into the RCS file, deleting the existing text.

−T Preserve the modification time on the RCS file unless a revision is removed. This option can suppress extensive recompilation caused by a **make**(1) dependency of some copy of the working file on the RCS file. Use this option with care; it can suppress recompilation even when it is needed, i.e. when a change to the RCS file would mean a change to keyword strings in the working file.

−V Print RCS's version number.

−V*n* Emulate RCS version *n*. See **co**(1) for details.

−x*suffixes*
Use *suffixes* to characterize RCS files. See **ci**(1) for details.

−z*zone* Use *zone* as the default time zone. This option has no effect; it is present for compatibility with other RCS commands.

At least one explicit option must be given, to ensure compatibility with future planned extensions to the **rcs** command.

COMPATIBILITY

The **−b***rev* option generates an RCS file that cannot be parsed by RCS version 3 or earlier.

The **−k***subst* options (except **−kkv**) generate an RCS file that cannot be parsed by RCS version 4 or earlier.

Use **rcs −V***n* to make an RCS file acceptable to RCS version *n* by discarding information that would confuse version *n*.

RCS version 5.5 and earlier does not support the **−x** option, and requires a **,v** suffix on an RCS file name.

FILES

rcs accesses files much as **ci**(1) does, except that it uses the effective user for all accesses, it does not write the working file or its directory, and it does not even read the working file unless a revision number of **$** is specified.

ENVIRONMENT

RCSINIT

Options prepended to the argument list, separated by spaces. A backslash escapes spaces within an option. The **RCSINIT** options are prepended to the argument lists of most RCS commands. Useful **RCSINIT** options include **−q**, **−V**, **−x**, and **−z**.

RCS_MEM_LIMIT

An integer *lim*, measured in kilobytes, specifying the threshold under which commands will try to use memory-based operations for processing the RCS file. (For RCS files of size *lim* kilobytes or greater, RCS will use the slower standard input/output routines.) Default value is 256.

TMPDIR

Name of the temporary directory. If not set, the environment variables **TMP** and **TEMP** are inspected instead and the first value found is taken; if none of them are set, a host-dependent default is used, typically **/tmp**.

DIAGNOSTICS

The RCS file name and the revisions outdated are written to the diagnostic output. The exit status is zero if and only if all operations were successful.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.8; Release Date: 2011-08-30.

SEE ALSO

rcsintro(1), **co**(1), **ci**(1), **ident**(1), **rcsclean**(1), **rcsdiff**(1), **rcsmerge**(1), **rlog**(1), **rcsfile**(5).

Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

BUGS

A catastrophe (e.g. a system crash) can cause RCS to leave behind a semaphore file that causes later invocations of RCS to claim that the RCS file is in use. To fix this, remove the semaphore file. A semaphore file's name typically begins with **,** or ends with **_**.

The separator for revision ranges in the **−o** option used to be − instead of **:**, but this leads to confusion when symbolic names contain −. For backwards compatibility **rcs −o** still supports the old − separator, but it warns about this obsolete use.

Symbolic names need not refer to existing revisions or branches. For example, the **−o** option does not

remove symbolic names for the outdated revisions; you must use **−n** to remove the names.

NAME

ci − check in RCS revisions

SYNOPSIS

ci [*options*] *file* . . .

DESCRIPTION

ci stores new revisions into RCS files. Each file name matching an RCS suffix is taken to be an RCS file. All others are assumed to be working files containing new revisions. **ci** deposits the contents of each working file into the corresponding RCS file. If only a working file is given, **ci** tries to find the corresponding RCS file in an RCS subdirectory and then in the working file's directory. For more details, see FILE NAMING below.

For **ci** to work, the caller's login must be on the access list, except if the access list is empty or the caller is the superuser or the owner of the file. To append a new revision to an existing branch, the tip revision on that branch must be locked by the caller. Otherwise, only a new branch can be created. This restriction is not enforced for the owner of the file if non-strict locking is used (see **rcs**(1)). A lock held by someone else can be broken with the **rcs** command.

Unless the **−f** option is given, **ci** checks whether the revision to be deposited differs from the preceding one. If not, instead of creating a new revision **ci** reverts to the preceding one. To revert, ordinary **ci** removes the working file and any lock; **ci −l** keeps and **ci −u** removes any lock, and then they both generate a new working file much as if **co −l** or **co −u** had been applied to the preceding revision. When reverting, any **−n** and **−s** options apply to the preceding revision.

For each revision deposited, **ci** prompts for a log message. The log message should summarize the change and must be terminated by end-of-file or by a line containing **.** by itself. If several files are checked in **ci** asks whether to reuse the previous log message. If the standard input is not a terminal, **ci** suppresses the prompt and uses the same log message for all files. See also **−m**.

If the RCS file does not exist, **ci** creates it and deposits the contents of the working file as the initial revision (default number: **1.1**). The access list is initialized to empty. Instead of the log message, **ci** requests descriptive text (see **−t** below).

The number *rev* of the deposited revision can be given by any of the options **−f**, **−i**, **−I**, **−j**, **−k**, **−l**, **−M**, **−q**, **−r**, or **−u**. *rev* can be symbolic, numeric, or mixed. Symbolic names in *rev* must already be defined; see the **−n** and **−N** options for assigning names during checkin. If *rev* is **$**, **ci** determines the revision number from keyword values in the working file.

If *rev* begins with a period, then the default branch (normally the trunk) is prepended to it. If *rev* is a branch number followed by a period, then the latest revision on that branch is used.

If *rev* is a revision number, it must be higher than the latest one on the branch to which *rev* belongs, or must start a new branch.

If *rev* is a branch rather than a revision number, the new revision is appended to that branch. The level number is obtained by incrementing the tip revision number of that branch. If *rev* indicates a non-existing branch, that branch is created with the initial revision numbered *rev***.1**.

If *rev* is omitted, **ci** tries to derive the new revision number from the caller's last lock. If the caller has locked the tip revision of a branch, the new revision is appended to that branch. The new revision number is obtained by incrementing the tip revision number. If the caller locked a non-tip revision, a new branch is started at that revision by incrementing the highest branch number at that revision. The default initial branch and level numbers are **1**.

If *rev* is omitted and the caller has no lock, but owns the file and locking is not set to *strict*, then the revision is appended to the default branch (normally the trunk; see the **−b** option of **rcs**(1)).

Exception: On the trunk, revisions can be appended to the end, but not inserted.

OPTIONS

−r*rev* Check in revision *rev*.

−r The bare **−r** option (without any revision) has an unusual meaning in **ci**. With other RCS commands, a bare **−r** option specifies the most recent revision on the default branch, but with **ci**, a bare **−r** option reestablishes the default behavior of releasing a lock and removing the working file, and is used to override any default **−l** or **−u** options established by shell aliases or scripts.

−l[*rev*] works like **−r**, except it performs an additional **co −l** for the deposited revision. Thus, the deposited revision is immediately checked out again and locked. This is useful for saving a revision although one wants to continue editing it after the checkin.

−u[*rev*] works like **−l**, except that the deposited revision is not locked. This lets one read the working file immediately after checkin.

The **−l**, bare **−r**, and **−u** options are mutually exclusive and silently override each other. For example, **ci −u −r** is equivalent to **ci −r** because bare **−r** overrides **−u**.

−f[*rev*] forces a deposit; the new revision is deposited even it is not different from the preceding one.

−k[*rev*] searches the working file for keyword values to determine its revision number, creation date, state, and author (see **co**(1)), and assigns these values to the deposited revision, rather than computing them locally. It also generates a default login message noting the login of the caller and the actual checkin date. This option is useful for software distribution. A revision that is sent to several sites should be checked in with the **−k** option at these sites to preserve the original number, date, author, and state. The extracted keyword values and the default log message can be overridden with the options **−d**, **−m**, **−s**, **−w**, and any option that carries a revision number.

−q[*rev*] quiet mode; diagnostic output is not printed. A revision that is not different from the preceding one is not deposited, unless **−f** is given.

−i[*rev*] initial checkin; report an error if the RCS file already exists. This avoids race conditions in certain applications.

−j[*rev*] just checkin and do not initialize; report an error if the RCS file does not already exist.

−I[*rev*] interactive mode; the user is prompted and questioned even if the standard input is not a terminal.

−d[*date*]
uses *date* for the checkin date and time. The *date* is specified in free format as explained in **co**(1). This is useful for lying about the checkin date, and for **−k** if no date is available. If *date* is empty, the working file's time of last modification is used.

−M[*rev*]
Set the modification time on any new working file to be the date of the retrieved revision. For example, **ci −d −M −u** *f* does not alter *f*'s modification time, even if *f*'s contents change due to keyword substitution. Use this option with care; it can confuse **make**(1).

−m[*msg***]**
uses the string *msg* as the log message for all revisions checked in. If *msg* is omitted, it defaults to "*** empty log message ***". By convention, log messages that start with **#** are comments and are ignored by programs like GNU Emacs's **vc** package. Also, log messages that start with **{***clumpname***}** (followed by white space) are meant to be clumped together if possible, even if they are associated with different files; the **{***clumpname***}** label is used only for clumping, and is not considered to be part of the log message itself.

−n*name*
assigns the symbolic name *name* to the number of the checked-in revision. **ci** prints an error message if *name* is already assigned to another number.

−N*name*
same as **−n**, except that it overrides a previous assignment of *name*.

−s*state* sets the state of the checked-in revision to the identifier *state*. The default state is **Exp**.

−t *file* writes descriptive text from the contents of the named *file* into the RCS file, deleting the existing text. The *file* cannot begin with **−**.

−t−*string*

Write descriptive text from the *string* into the RCS file, deleting the existing text.

The **−t** option, in both its forms, has effect only during an initial checkin; it is silently ignored otherwise.

During the initial checkin, if **−t** is not given, **ci** obtains the text from standard input, terminated by end-of-file or by a line containing **.** by itself. The user is prompted for the text if interaction is possible; see **−I**.

For backward compatibility with older versions of RCS, a bare **−t** option is ignored.

−T Set the RCS file's modification time to the new revision's time if the former precedes the latter and there is a new revision; preserve the RCS file's modification time otherwise. If you have locked a revision, **ci** usually updates the RCS file's modification time to the current time, because the lock is stored in the RCS file and removing the lock requires changing the RCS file. This can create an RCS file newer than the working file in one of two ways: first, **ci −M** can create a working file with a date before the current time; second, when reverting to the previous revision the RCS file can change while the working file remains unchanged. These two cases can cause excessive recompilation caused by a **make**(1) dependency of the working file on the RCS file. The **−T** option inhibits this recompilation by lying about the RCS file's date. Use this option with care; it can suppress recompilation even when a checkin of one working file should affect another working file associated with the same RCS file. For example, suppose the RCS file's time is 01:00, the (changed) working file's time is 02:00, some other copy of the working file has a time of 03:00, and the current time is 04:00. Then **ci −d −T** sets the RCS file's time to 02:00 instead of the usual 04:00; this causes **make**(1) to think (incorrectly) that the other copy is newer than the RCS file.

−w*login*

uses *login* for the author field of the deposited revision. Useful for lying about the author, and for **−k** if no author is available.

−V Print RCS's version number.

−V*n* Emulate RCS version *n*. See **co**(1) for details.

−x*suffixes*

specifies the suffixes for RCS files. A nonempty suffix matches any file name ending in the suffix. An empty suffix matches any file name of the form **RCS/** *frag* or *frag1*/**RCS/** *frag2*. The **−x** option can specify a list of suffixes separated by /. For example, **−x,v/** specifies two suffixes: **,v** and the empty suffix. If two or more suffixes are specified, they are tried in order when looking for an RCS file; the first one that works is used for that file. If no RCS file is found but an RCS file can be created, the suffixes are tried in order to determine the new RCS file's name. The default for *suffixes* is installation-dependent; normally it is **,v/** for hosts like Unix that permit commas in file names, and is empty (i.e. just the empty suffix) for other hosts.

−z*zone* specifies the date output format in keyword substitution, and specifies the default time zone for *date* in the **−d***date* option. The *zone* should be empty, a numeric UTC offset, or the special string **LT** for local time. The default is an empty *zone*, which uses the traditional RCS format of UTC without any time zone indication and with slashes separating the parts of the date; otherwise, times are output in ISO 8601 format with time zone indication. For example, if local time is January 11, 1990, 8pm Pacific Standard Time, eight hours west of UTC, then the time is output as follows:

option	*time output*	
−z	**1990/01/12 04:00:00**	*(default)*
−zLT	**1990-01-11 20:00:00−08**	
−z+05:30	**1990-01-12 09:30:00+05:30**	

The **−z** option does not affect dates stored in RCS files, which are always UTC.

FILE NAMING

Pairs of RCS files and working files can be specified in three ways (see also the example section).

1) Both the RCS file and the working file are given. The RCS file name is of the form *frag1*/*workfileX* and the working file name is of the form *frag2*/*workfile* where *frag1*/ and *frag2*/ are (possibly different or empty) file names, *workfile* is a file name, and *X* is an RCS suffix. If *X* is empty, *frag1*/ must start with **RCS/** or must contain **/RCS/**.

2) Only the RCS file is given. Then the working file is created in the current directory and its name is derived from the RCS file name by removing *frag1*/ and the suffix *X*.

3) Only the working file is given. Then **ci** considers each RCS suffix *X* in turn, looking for an RCS file of the form *frag2*/**RCS**/*workfileX* or (if the former is not found and *X* is nonempty) *frag2*/*workfileX*.

If the RCS file is specified without a file name in 1) and 2), **ci** looks for the RCS file first in the directory **./RCS** and then in the current directory.

ci reports an error if an attempt to open an RCS file fails for an unusual reason, even if the RCS file's name is just one of several possibilities. For example, to suppress use of RCS commands in a directory *d*, create a regular file named *d*/**RCS** so that casual attempts to use RCS commands in *d* fail because *d*/**RCS** is not a directory.

EXAMPLES

Suppose **,v** is an RCS suffix and the current directory contains a subdirectory **RCS** with an RCS file **io.c,v**. Then each of the following commands check in a copy of **io.c** into **RCS/io.c,v** as the latest revision, removing **io.c**.

```
ci io.c;   ci RCS/io.c,v;   ci io.c,v;
ci io.c RCS/io.c,v;   ci io.c io.c,v;
ci RCS/io.c,v io.c;   ci io.c,v io.c;
```

Suppose instead that the empty suffix is an RCS suffix and the current directory contains a subdirectory **RCS** with an RCS file **io.c**. The each of the following commands checks in a new revision.

```
ci io.c;   ci RCS/io.c;
ci io.c RCS/io.c;
ci RCS/io.c io.c;
```

FILE MODES

An RCS file created by **ci** inherits the read and execute permissions from the working file. If the RCS file exists already, **ci** preserves its read and execute permissions. **ci** always turns off all write permissions of RCS files.

FILES

Temporary files are created in the directory containing the working file, and also in the temporary directory (see **TMPDIR** under **ENVIRONMENT**). A semaphore file or files are created in the directory containing the RCS file. With a nonempty suffix, the semaphore names begin with the first character of the suffix; therefore, do not specify an suffix whose first character could be that of a working file name. With an empty suffix, the semaphore names end with **_** so working file names should not end in **_**.

ci never changes an RCS file or working file. Normally, **ci** unlinks the file and creates a new one; but instead of breaking a chain of one or more symbolic links to an RCS file, it unlinks the destination file instead. Therefore, **ci** breaks any hard or symbolic links to any working file it changes; and hard links to RCS files are ineffective, but symbolic links to RCS files are preserved.

The effective user must be able to search and write the directory containing the RCS file. Normally, the real user must be able to read the RCS and working files and to search and write the directory containing the working file; however, some older hosts cannot easily switch between real and effective users, so on these hosts the effective user is used for all accesses. The effective user is the same as the real user unless your copies of **ci** and **co** have setuid privileges. As described in the next section, these privileges yield extra security if the effective user owns all RCS files and directories, and if only the effective user can write RCS directories.

Users can control access to RCS files by setting the permissions of the directory containing the files; only users with write access to the directory can use RCS commands to change its RCS files. For example, in

hosts that allow a user to belong to several groups, one can make a group's RCS directories writable to that group only. This approach suffices for informal projects, but it means that any group member can arbitrarily change the group's RCS files, and can even remove them entirely. Hence more formal projects sometimes distinguish between an RCS administrator, who can change the RCS files at will, and other project members, who can check in new revisions but cannot otherwise change the RCS files.

SETUID USE

To prevent anybody but their RCS administrator from deleting revisions, a set of users can employ setuid privileges as follows.

- Check that the host supports RCS setuid use. Consult a trustworthy expert if there are any doubts. It is best if the **seteuid** system call works as described in Posix 1003.1a Draft 5, because RCS can switch back and forth easily between real and effective users, even if the real user is **root**. If not, the second best is if the **setuid** system call supports saved setuid (the {_POSIX_SAVED_IDS} behavior of Posix 1003.1-1990); this fails only if the real or effective user is **root**. If RCS detects any failure in setuid, it quits immediately.
- Choose a user *A* to serve as RCS administrator for the set of users. Only *A* can invoke the **rcs** command on the users' RCS files. *A* should not be **root** or any other user with special powers. Mutually suspicious sets of users should use different administrators.
- Choose a file name *B* to be a directory of files to be executed by the users.
- Have *A* set up *B* to contain copies of **ci** and **co** that are setuid to *A* by copying the commands from their standard installation directory *D* as follows:

  mkdir *B*
  cp *D*/**c[io]** *B*
  chmod go−w,u+s *B*/**c[io]**

- Have each user prepend *B* to their command search path as follows:

  PATH=*B***:$PATH; export PATH** # ordinary shell
  set path=(*B* **$path)** # C shell

- Have *A* create each RCS directory *R* with write access only to *A* as follows:

  mkdir *R*
  chmod go−w *R*

- If you want to let only certain users read the RCS files, put the users into a group *G*, and have *A* further protect the RCS directory as follows:

  chgrp *G R*
  chmod g−w,o−rwx *R*

- Have *A* copy old RCS files (if any) into *R*, to ensure that *A* owns them.
- An RCS file's access list limits who can check in and lock revisions. The default access list is empty, which grants checkin access to anyone who can read the RCS file. If you want limit checkin access, have *A* invoke **rcs −a** on the file; see **rcs**(1). In particular, **rcs −e −a***A* limits access to just *A*.
- Have *A* initialize any new RCS files with **rcs −i** before initial checkin, adding the **−a** option if you want to limit checkin access.
- Give setuid privileges only to **ci**, **co**, and **rcsclean**; do not give them to **rcs** or to any other command.
- Do not use other setuid commands to invoke RCS commands; setuid is trickier than you think!

ENVIRONMENT

RCSINIT

Options prepended to the argument list, separated by spaces. A backslash escapes spaces within an option. The **RCSINIT** options are prepended to the argument lists of most RCS commands. Useful **RCSINIT** options include **−q**, **−V**, **−x**, and **−z**.

RCS_MEM_LIMIT

An integer *lim*, measured in kilobytes, specifying the threshold under which commands will try to use memory-based operations for processing the RCS file. (For RCS files of size *lim* kilobytes or greater, RCS will use the slower standard input/output routines.) Default value is 256.

TMPDIR

Name of the temporary directory. If not set, the environment variables **TMP** and **TEMP** are inspected instead and the first value found is taken; if none of them are set, a host-dependent default is used, typically **/tmp**.

DIAGNOSTICS

For each revision, **ci** prints the RCS file, the working file, and the number of both the deposited and the preceding revision. The exit status is zero if and only if all operations were successful.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.8; Release Date: 2011-08-30.

SEE ALSO

co(1), **emacs**(1), **ident**(1), **make**(1), **rcs**(1), **rcsclean**(1), **rcsdiff**(1), **rcsintro**(1), **rcsmerge**(1), **rlog**(1), **setuid**(2), **rcsfile**(5).

Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

NAME

rcsdiff − compare RCS revisions

SYNOPSIS

rcsdiff [**−k***subst*] [**−q**] [**−r***rev1* [**−r***rev2*]] [**−T**] [**−V**[*n*]] [**−x***suffixes*] [**−z***zone*] [*diff options*] *file* ...

DESCRIPTION

rcsdiff runs **diff**(1) to compare two revisions of each RCS file given.

Filenames matching an RCS suffix denote RCS files; all others denote working files. Names are paired as explained in **ci**(1).

The option **−q** suppresses diagnostic output. Zero, one, or two revisions may be specified with **−r**. The option **−k***subst* affects keyword substitution when extracting revisions, as described in **co**(1); for example, **−kk −r1.1 −r1.2** ignores differences in keyword values when comparing revisions **1.1** and **1.2**. To avoid excess output from locker name substitution, **−kkvl** is assumed if (1) at most one revision option is given, (2) no **−k** option is given, (3) **−kkv** is the default keyword substitution, and (4) the working file's mode would be produced by **co −l**. See **co**(1) for details about **−T**, **−V**, **−x** and **−z**. Otherwise, all options of **diff**(1) that apply to regular files are accepted, with the same meaning as for **diff**.

If both *rev1* and *rev2* are omitted, **rcsdiff** compares the latest revision on the default branch (by default the trunk) with the contents of the corresponding working file. This is useful for determining what you changed since the last checkin.

If *rev1* is given, but *rev2* is omitted, **rcsdiff** compares revision *rev1* of the RCS file with the contents of the corresponding working file.

If both *rev1* and *rev2* are given, **rcsdiff** compares revisions *rev1* and *rev2* of the RCS file.

Both *rev1* and *rev2* may be given numerically or symbolically.

EXAMPLE

The command

rcsdiff f.c

compares the latest revision on the default branch of the RCS file to the contents of the working file **f.c**.

ENVIRONMENT

RCSINIT

Options prepended to the argument list, separated by spaces. A backslash escapes spaces within an option. The **RCSINIT** options are prepended to the argument lists of most RCS commands. Useful **RCSINIT** options include **−q**, **−V**, **−x**, and **−z**.

RCS_MEM_LIMIT

An integer *lim*, measured in kilobytes, specifying the threshold under which commands will try to use memory-based operations for processing the RCS file. (For RCS files of size *lim* kilobytes or greater, RCS will use the slower standard input/output routines.) Default value is 256.

TMPDIR

Name of the temporary directory. If not set, the environment variables **TMP** and **TEMP** are inspected instead and the first value found is taken; if none of them are set, a host-dependent default is used, typically **/tmp**.

DIAGNOSTICS

Exit status is 0 for no differences during any comparison, 1 for some differences, 2 for trouble.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.8; Release Date: 2011-08-30.

SEE ALSO

ci(1), **co**(1), **diff**(1), **ident**(1), **rcs**(1), **rcsintro**(1), **rcsmerge**(1), **rlog**(1).

Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

NAME

biosdecode − BIOS information decoder

SYNOPSIS

biosdecode [**OPTIONS**]

DESCRIPTION

biosdecode parses the BIOS memory and prints information about all structures (or entry points) it knows of. Currently known entry point types are:

- SMBIOS (System Management BIOS)
  Use **dmidecode** for a more detailed output.
- DMI (Desktop Management Interface, a legacy version of SMBIOS)
  Use **dmidecode** for a more detailed output.
- SYSID
- PNP (Plug and Play)
- ACPI (Advanced Configuration and Power Interface)
- BIOS32 (BIOS32 Service Directory)
- PIR (PCI IRQ Routing)
- 32OS (BIOS32 Extension, Compaq-specific)
  See **ownership** for a Compaq ownership tag retrieval tool.
- SNY (Sony-specific, not decoded)
- VPD (Vital Product Data, IBM-specific)
  Use **vpddecode** for a more detailed output.
- FJKEYINF (Application Panel, Fujitsu-specific)

biosdecode started its life as a part of **dmidecode** but as more entry point types were added, if was moved to a different program.

OPTIONS

-d, **--dev-mem FILE**
Read memory from device **FILE** (default: **/dev/mem**)

-h, **--help**
Display usage information and exit

-V, **--version**
Display the version and exit

FILES

/dev/mem

BUGS

Most of the time, **biosdecode** prints too much information (you don't really care about addresses) or not enough (because it doesn't follow pointers and has no lookup tables).

AUTHORS

Alan Cox, Jean Delvare

SEE ALSO

dmidecode(8), **mem**(4), **ownership**(8), **vpddecode**(8)

NAME

vpddecode − VPD structure decoder

SYNOPSIS

vpddecode [**OPTIONS**]

DESCRIPTION

vpddecode prints the "vital product data" information that can be found in almost all IBM and Lenovo computers. Available items are:

- BIOS Build ID
- Box Serial Number
- Motherboard Serial Number
- Machine Type/Model

Some systems have these additional items:

- BIOS Release Date
- Default Flash Image File Name

Note that these additional items are not documented by IBM, so this is guess work, and as such should not be blindly trusted. Feedback about the accuracy of these labels is welcome.

OPTIONS

-d, **--dev-mem FILE**
Read memory from device **FILE** (default: **/dev/mem**)

-s, **--string KEYWORD**
Only display the value of the VPD string identified by **KEYWORD**. **KEYWORD** must be a keyword from the following list: **bios-build-id**, **box-serial-number**, **motherboard-serial-number**, **machine-type-model**, **bios-release-date**. Each keyword corresponds to an offset and a length within the VPD record. Not all strings may be defined on all VPD-enabled systems. If **KEYWORD** is not provided or not valid, a list of all valid keywords is printed and **vpddecode** exits with an error. This option cannot be used more than once. Mutually exclusive with **--dump**.

-u, **--dump**
Do not decode the VPD records, dump their contents as hexadecimal instead. Note that this is still a text output, no binary data will be thrown upon you. ASCII equivalent is displayed when possible. This option is mainly useful for debugging. Mutually exclusive with **--string**.

-h, **--help**
Display usage information and exit

-V, **--version**
Display the version and exit

FILES

/dev/mem

AUTHOR

Jean Delvare

SEE ALSO

biosdecode(8), **dmidecode**(8), **mem**(4), **ownership**(8)

NAME

dmidecode − DMI table decoder

SYNOPSIS

dmidecode [**OPTIONS**]

DESCRIPTION

dmidecode is a tool for dumping a computer's DMI (some say SMBIOS) table contents in a human-readable format. This table contains a description of the system's hardware components, as well as other useful pieces of information such as serial numbers and BIOS revision. Thanks to this table, you can retrieve this information without having to probe for the actual hardware. While this is a good point in terms of report speed and safeness, this also makes the presented information possibly unreliable.

The DMI table doesn't only describe what the system is currently made of, it also can report the possible evolutions (such as the fastest supported CPU or the maximal amount of memory supported).

SMBIOS stands for System Management BIOS, while DMI stands for Desktop Management Interface. Both standards are tightly related and developed by the DMTF (Desktop Management Task Force).

As you run it, **dmidecode** will try to locate the DMI table. If it succeeds, it will then parse this table and display a list of records like this one:

Handle 0x0002, DMI type 2, 8 bytes. Base Board Information
    Manufacturer: Intel
    Product Name: C440GX+
    Version: 727281-001
    Serial Number: INCY92700942

Each record has:

- A handle. This is a unique identifier, which allows records to reference each other. For example, processor records usually reference cache memory records using their handles.
- A type. The SMBIOS specification defines different types of elements a computer can be made of. In this example, the type is 2, which means that the record contains "Base Board Information".
- A size. Each record has a 4-byte header (2 for the handle, 1 for the type, 1 for the size), the rest is used by the record data. This value doesn't take text strings into account (these are placed at the end of the record), so the actual length of the record may be (and is often) greater than the displayed value.
- Decoded values. The information presented of course depends on the type of record. Here, we learn about the board's manufacturer, model, version and serial number.

OPTIONS

-d, **--dev-mem FILE**
Read memory from device **FILE** (default: **/dev/mem**)

-q, **--quiet**
Be less verbose. Unknown, inactive and OEM-specific entries are not displayed. Meta-data and handle references are hidden.

-s, **--string KEYWORD**
Only display the value of the DMI string identified by **KEYWORD**. **KEYWORD** must be a keyword from the following list: **bios-vendor**, **bios-version**, **bios-release-date**, **system-manufacturer**, **system-product-name**, **system-version**, **system-serial-number**, **system-uuid**, **baseboard-manufacturer**, **baseboard-product-name**, **baseboard-version**, **baseboard-serial-number**, **baseboard-asset-tag**, **chassis-manufacturer**, **chassis-type**, **chassis-version**, **chassis-serial-number**, **chassis-asset-tag**, **processor-family**, **processor-manufacturer**, **processor-version**,

processor-frequency. Each keyword corresponds to a given DMI type and a given offset within this entry type. Not all strings may be meaningful or even defined on all systems. Some keywords may return more than one result on some systems (e.g. **processor-version** on a multi-processor system). If **KEYWORD** is not provided or not valid, a list of all valid keywords is printed and **dmidecode** exits with an error. This option cannot be used more than once.

-t, **--type TYPE**
Only display the entries of type **TYPE**. **TYPE** can be either a DMI type number, or a comma-separated list of type numbers, or a keyword from the following list: **bios**, **system**, **baseboard**, **chassis**, **processor**, **memory**, **cache**, **connector**, **slot**. Refer to the DMI TYPES section below for details. If this option is used more than once, the set of displayed entries will be the union of all the given types. If **TYPE** is not provided or not valid, a list of all valid keywords is printed and **dmidecode** exits with an error.

-u, **--dump**
Do not decode the entries, dump their contents as hexadecimal instead. Note that this is still a text output, no binary data will be thrown upon you. The strings attached to each entry are displayed as both hexadecimal and ASCII. This option is mainly useful for debugging.

--dump-bin FILE
Do not decode the entries, instead dump the DMI data to a file in binary form. The generated file is suitable to pass to **--from-dump** later.

--from-dump FILE
Read the DMI data from a binary file previously generated using **--dump-bin**.

-h, **--help**
Display usage information and exit

-V, **--version**
Display the version and exit

Options --string, --type and --dump-bin determine the output format and are mutually exclusive.

Please note in case of **dmidecode** is run on a system with BIOS that boasts new SMBIOS specification, which is not supported by the tool yet, it will print out relevant message in addition to requested data on the very top of the output. Thus informs the output data is not reliable.

DMI TYPES

The SMBIOS specification defines the following DMI types:

r l __ r l. Type Information 0 BIOS 1 System 2 Baseboard 3 Chassis 4 Processor 5 Memory Controller 6 Memory Module 7 Cache 8 Port Connector 9 System Slots 10 On Board Devices 11 OEM Strings 12 System Configuration Options 13 BIOS Language 14 Group Associations 15 System Event Log 16 Physical Memory Array 17 Memory Device 18 32-bit Memory Error 19 Memory Array Mapped Address 20 Memory Device Mapped Address 21 Built-in Pointing Device 22 Portable Battery 23 System Reset 24 Hardware Security 25 System Power Controls 26 Voltage Probe 27 Cooling Device 28 Temperature Probe 29 Electrical Current Probe 30 Out-of-band Remote Access 31 Boot Integrity Services 32 System Boot 33 64-bit Memory Error 34 Management Device 35 Management Device Component 36 Management Device Threshold Data 37 Memory Channel 38 IPMI Device 39 Power Supply 40 Additional Information 41 Onboard Devices Extended Information 42 Management Controller Host Interface

Additionally, type 126 is used for disabled entries and type 127 is an end-of-table marker. Types 128 to 255 are for OEM-specific data. **dmidecode** will display these entries by default, but it can only decode them when the vendors have contributed documentation or code for them.

Keywords can be used instead of type numbers with **--type**. Each keyword is equivalent to a list of type numbers:

l l __ l l. Keyword Types bios 0, 13 system 1, 12, 15, 23, 32 baseboard 2, 10, 41 chassis 3 processor 4 memory 5, 6, 16, 17 cache 7 connector 8 slot 9

Keywords are matched case-insensitively. The following command lines are equivalent:

- dmidecode --type 0 --type 13
- dmidecode --type 0,13
- dmidecode --type bios
- dmidecode --type BIOS

BINARY DUMP FILE FORMAT

The binary dump files generated by --dump-bin and read using --from-dump are formatted as follows:

- The SMBIOS or DMI entry point is located at offset 0x00. It is crafted to hard-code the table address at offset 0x20.
- The DMI table is located at offset 0x20.

FILES

/dev/mem

BUGS

More often than not, information contained in the DMI tables is inaccurate, incomplete or simply wrong.

AUTHORS

Alan Cox, Jean Delvare

SEE ALSO

biosdecode(8), **mem**(4), **ownership**(8), **vpddecode**(8)

NAME

ownership − Compaq ownership tag retriever

SYNOPSIS

ownership [**OPTIONS**]

DESCRIPTION

ownership retrieves and prints the "ownership tag" that can be set on Compaq computers. Contrary to all other programs of the **dmidecode** package, **ownership** doesn't print any version information, nor labels, but only the raw ownership tag. This should help its integration in scripts.

OPTIONS

-d, **--dev-mem FILE**
Read memory from device **FILE** (default: **/dev/mem**)

-h, **--help**
Display usage information and exit

-V, **--version**
Display the version and exit

FILES

/dev/mem

AUTHOR

Jean Delvare

SEE ALSO

biosdecode(8), **dmidecode**(8), **mem**(4), **vpddecode**(8)

NAME

statgrab−make−mrtg−config − generates MRTG configuration

SYNOPSIS

statgrab−make−mrtg−config [OPTION]...

DESCRIPTION

statgrab−make−mrtg−config generates MRTG configuration from statgrab output and writes it to stdout.

OPTIONS

The following options are supported.

−−no−header
: Don’t print MRTG global options; useful if you want to include the output of this script in another MRTG config file

−−workdir *PATH*
: Use PATH for MRTG’s WorkDir option

−−statgrab *PATH*
: Specify location of statgrab binary (default "statgrab")

−−help Display help and exit

SEE ALSO

statgrab(1) **statgrab−make−mrtg−index**(1)

AUTHORS

This man page was derived from the man page written by Bartosz Fenski for the Debian GNU/Linux distribution.

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_fs_stats − get filesystem statistics

SYNOPSIS

#include <statgrab.h>

sg_fs_stats *sg_get_fs_stats(int *entries);

DESCRIPTION

The sg_get_fs_stats takes a pointer to an int, entries, which is filled with the number of mounted file systems the machine has. The return value is a pointer to the first member of an array of sg_fs_stats structures; the number of entries in the array is returned in entries.

The function returns statistics about mounted filesystems, including free space and inode usage.

RETURN VALUES

sg_get_fs_stats returns a pointer to a structure of type sg_fs_stats.

```
typedef struct {
    char *device_name;
    char *fs_type;
    char *mnt_point;
    long long size;
    long long used;
    long long avail;
    long long total_inodes;
    long long used_inodes;
    long long free_inodes;
    long long avail_inodes;
    long long io_size;
    long long block_size;
    long long total_blocks;
    long long free_blocks;
    long long used_blocks;
    long long avail_blocks;
} sg_fs_stats;
```

device_name
: The name known to the operating system. (eg. on linux it might be hda)

fs_type
: The type of the filesystem.

mnt_point
: The mount point of the file system.

size
: The size, in bytes, of the file system.

used
: The amount of space, in bytes, used on the filesystem.

avail
: The amount of space, in bytes, available on the filesystem.

total_inodes
: The total number of inodes in the filesystem.

used_inodes
: The number of used inodes in the filesystem.

free_inodes
: The number of free inodes in the filesystem.

avail_inodes
: The number of free inodes available to non−privileged processes.

io_size
: A suggested optimal block size for IO operations −− if you’re reading or writing lots of data, do it in chunks of this size.

block_size
: How big blocks actually are on the underlying filesystem (typically for purposes of stats reporting).

total_blocks
: The total number of blocks in the filesystem.

free_blocks
: The number of free blocks in the filesystem.

used_blocks
: The number of used blocks in the filesystem.

avail_blocks
: The number of free blocks available to non−privileged processes.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_mem_stats, sg_get_swap_stats − get VM statistics

SYNOPSIS

#include <statgrab.h>

sg_mem_stats *sg_get_mem_stats(void);

sg_swap_stats *sg_get_swap_stats(void);

DESCRIPTION

Memory statistics are accessed through the sg_get_mem_stats function. It returns a pointer to a static sg_mem_stats.

The sg_get_swap_stats returns returns swap statistics. It returns a pointer to a static sg_swap_stats.

On the FreeBSD operating system elevated privileges are required to access the swap statistics. Making the program setgid kmem should be sufficient. Programs running as root will not have this problem.

RETURN VALUES

The VM system calls can return a pointer to either a sg_mem_stats or a sg_swap_stats.

```
typedef struct{
    long long total;
    long long free;
    long long used;
    long long cache;
}sg_mem_stats;
```

total	The total amount of memory in bytes.
free	The total free memory in bytes.
used	The total used memory in bytes.
cache	The amount of cache used in bytes.

```
typedef struct{
    long long total;
    long long used;
    long long free;
}sg_swap_stats;
```

total	The total swap space in bytes.
used	The used swap in bytes.
free	The free swap in bytes.

TODO

Add a function to hold open the file descriptor to the kernel memory structures. Doing this would allow the elevated privileges to be dropped early on.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_process_stats, sg_get_process_count, sg_process_compare_name, sg_process_compare_pid, sg_process_compare_uid, sg_process_compare_gid, sg_process_compare_size, sg_process_compare_res, sg_process_compare_cpu, sg_process_compare_time − get process statistics

SYNOPSIS

#include <statgrab.h>

sg_process_stats *sg_get_process_stats(int *entries);

sg_process_count *sg_get_process_count(void);

int sg_process_compare_name(const void *va, const void *vb);

int sg_process_compare_pid(const void *va, const void *vb);

int sg_process_compare_uid(const void *va, const void *vb);

int sg_process_compare_gid(const void *va, const void *vb);

int sg_process_compare_size(const void *va, const void *vb);

int sg_process_compare_res(const void *va, const void *vb);

int sg_process_compare_cpu(const void *va, const void *vb);

int sg_process_compare_time(const void *va, const void *vb);

DESCRIPTION

The sg_get_process_stats takes a pointer to an int, entries, which is filled with the number of processes the snapshot contains. The return value is a pointer to the first member of an array of sg_process_stats structures; the number of entries in the array is returned in entries.

sg_get_process_count returns a pointer to a static buffer of type sg_process_count.

These two functions provide information on the process running on the system. In the first case lots of detail is provided, and in the second case a summary of the states of the current processes.

sg_process_compare_name

sg_process_compare_pid

sg_process_compare_uid

sg_process_compare_gid

sg_process_compare_size

sg_process_compare_res

sg_process_compare_cpu

sg_process_compare_time

These functions compare two sg_process_stats entries, and return an int to represent which one is greater. The main use of these functions is to be passed to qsort to sort the sg_process_stats by the given type.

For example:

qsort(sg_process_stats ps, int ps_size, sizeof *ps, sg_process_compare_pid);

RETURN VALUES

The structure returned by sg_get_process_stats is of type sg_process_stats.

```
typedef struct {
    char *process_name;
    char *proctitle;
```

```
    pid_t pid;
    pid_t parent; /* Parent pid */
    pid_t pgid;   /* process id of process group leader */

    uid_t uid;
    uid_t euid;
    gid_t gid;
    gid_t egid;

    unsigned long long proc_size; /* in bytes */
    unsigned long long proc_resident; /* in bytes */
    time_t time_spent; /* time running in seconds */
    double cpu_percent;
    int nice;
    sg_process_state state;
} sg_process_stats;


typedef enum {
    SG_PROCESS_STATE_RUNNING,
    SG_PROCESS_STATE_SLEEPING,
    SG_PROCESS_STATE_STOPPED,
    SG_PROCESS_STATE_ZOMBIE,
    SG_PROCESS_STATE_UNKNOWN
} sg_process_state;
```

process_name
: The name of the command that was run.

proctitle
: The command line (the "title") of the process. Take note − this can be modifi ed by the process, so isn’t guaranteed to be the original command line.

pid
: The process ID.

parent
: The parent process ID.

pgid
: The process ID of the process group leader.

uid
: The UID the process is running as.

euid
: The effective UID the process is running as.

gid
: The GID the process is running as.

egid
: The effective GID the process is running as.

proc_size
: The size of the process in bytes.

proc_resident
: The size of the process that’s resident in memory.

time_spent
: The number of seconds the process has been running.

cpu_percent
: The current percentage of CPU the process is using.

nice
: The nice value of the process.

state The current state of the process. See sg_process_state for permitted values.

The structure returned by sg_get_process_count is of type sg_process_count.

```
typedef struct{
        int total;
        int running;
        int sleeping;
        int stopped;
        int zombie;
}sg_process_count;
```

total The total number of processes.

running
The number of running processes.

sleeping
The number of sleeping processes.

stopped
The number of stopped processes.

zombie The number of zombie processes.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_network_iface_stats − get network interface statistics

SYNOPSIS

#include <statgrab.h>

sg_network_iface_stats *_sg_get_network_iface_stats_(int *_entries_);

DESCRIPTION

The sg_get_network_iface_stats function takes a pointer to an int, entries, which is filled with the number of network interfaces the machine has. This is needed to know how many sg_network_iface_stats structures have been returned. A pointer is returned to the first sg_network_iface_stats.

sg_get_network_iface_stats returns statistics about the network interfaces in the machine. Specifically, it returns the speed of the interface, the duplex state, and whether it is currently up.

RETURN VALUES

The sg_get_network_iface_stats returns a pointer to a structure of type sg_network_iface_stats.

```
typedef enum{
    SG_IFACE_DUPLEX_FULL,
    SG_IFACE_DUPLEX_HALF,
    SG_IFACE_DUPLEX_UNKNOWN
}sg_iface_duplex;
```

Note: The SG_IFACE_DUPLEX_UNKNOWN value could mean that duplex hasn't been negotiated yet.

```
typedef struct{
    char *interface_name;
    int speed;
    sg_iface_duplex duplex;
    int up;
}sg_network_iface_stats;
```

interface_name
: The name known to the operating system. (eg. on linux it might be eth0)

speed
: The speed of the interface, in megabits/sec.

duplex
: The duplex state the interface is in. See sg_iface_duplex for permitted values.

up
: Whether the interface is up.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_cpu_stats, sg_get_cpu_stats_diff, sg_get_cpu_percents − get cpu usage

SYNOPSIS

#include <statgrab.h>

sg_cpu_percents *sg_get_cpu_percents(void);

sg_cpu_stats *sg_get_cpu_stats(void);

sg_cpu_stats *sg_get_cpu_stats_diff(void);

DESCRIPTION

sg_get_cpu_stats() and sg_get_cpu_stats_diff() both return static pointers of type sg_cpu_stats. sg_get_cpu_stats() returns the total amount of "ticks" the operating system has spent in each of the different states. sg_get_cpu_stats_diff() returns the difference in "ticks" for each of the states since last time sg_get_cpu_stats_diff() or sg_get_cpu_stats() was called. If it has never been called, it will return the result of sg_get_cpu_stats().

The value stored (the "ticks") will vary between operating systems. For example Solaris has a total of 100 per second, while Linux has substantially more. Also, different operating systems store different information − you won't find nice cpu on Solaris for example.

Because of this, you will ideally always want to work on a scale against the total, or in percentages.

sg_get_cpu_percents() returns a pointer to a static sg_cpu_percents. The function calls sg_get_cpu_stats_diff() and changes the values into percentages. If it has never been called before (and nor has sg_get_cpu_stats() or sg_get_cpu_stats_diff()), the returned percentages will be the systems total ever since its uptime. (Unless the counters have cycled)

RETURN VALUES

There are two structures returned by the CPU statistics functions.

```
typedef struct{
    long long user;
    long long kernel;
    long long idle;
    long long iowait;
    long long swap;
    long long nice;
    long long total;
    time_t systime;
}sg_cpu_stats;
```

```
typedef struct{
    float user;
    float kernel;
    float idle;
    float iowait;
    float swap;
    float nice;
    time_t time_taken;
}sg_cpu_percents;
```

user kernel idle iowait swap nice
: The different CPU states.

systime time_taken
The time taken in seconds since the last call of the function, or the system time.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

statgrab − get system statistics

SYNOPSIS

#include <statgrab.h>

int *sg_init*(void);

int *sg_drop_privileges*(void);

sg_error *sg_get_error*(void);

const char **sg_get_error_arg*(void);

int*sg_get_error_errno*(void);

const char **sg_str_error*(sg_error *code*);

sg_cpu_stats **sg_get_cpu_stats*(void);

sg_cpu_stats **sg_get_cpu_stats_diff*(void);

sg_cpu_percents **sg_get_cpu_percents*(void);

sg_disk_io_stats **sg_get_disk_io_stats*(int **entries*);

sg_disk_io_stats **sg_get_disk_io_stats_diff*(int **entries*);

sg_fs_stats **sg_get_fs_stats*(void);

sg_host_info **sg_get_host_info*(void);

sg_load_stats **sg_get_load_stats*(void);

sg_mem_stats **sg_get_mem_stats*(void);

sg_swap_stats **sg_get_swap_stats*(void);

sg_network_io_stats **sg_get_network_io_stats*(int **entries*);

sg_network_io_stats **sg_get_network_io_stats_diff*(int **entries*);

sg_network_iface_stats **sg_get_network_iface_stats*(int **entries*);

sg_page_stats **sg_get_page_stats*(void);

sg_page_stats **sg_get_page_stats_diff*(void);

sg_process_count **sg_get_process_stats*(void);

sg_user_stats **sg_get_user_stats*(void);

DESCRIPTION

The statgrab library provides a cross−platform interface to getting system statistics. Each of the function calls returns a structure containing statistics. See the manual page for each individual function for more details on usage.

sg_init must be the first function you call before you start to use libstatgrab; it performs all the one−time initialisation operations that need setuid/setgid privileges. For instance, on *BSD it opens a descriptor to be able to read kernel structures later on, and on Solaris it reads the device mappings that in some cases are only accessible by root (machines with a /dev/osa). Once this has run, the other libstatgrab functions no longer need elevated privileges. It is therefore a good idea to call sg_drop_privileges, which discards setuid and setgid privileges, immediately after you call sg_init, unless your application has another reason for needing setuid or setgid privileges.

sg_init and sg_drop_privileges return 0 on success, and non−zero on failure.

There are three functions relating to error reporting in libstatgrab. The first, sg_get_error returns an sg_error code which relates to the last error generated by libstatgrab. This can be converted to a string by calling sg_str_error giving the sg_error code as an argument. An optional argument may be set when the

error was generated. This can be accessed by calling sg_get_error_arg; NULL will be returned if no argument has been set. Some errors will also record the value of the system errno variable when the error occurred; this can be retrieved by calling sg_get_error_errno, which will return 0 if no valid errno has been recorded.

It is the intended practice that whenever a libstatgrab function is called and subsequently fails that an appropriate error will be set.

The library was originally written to support the i−scream central monitoring system, but has since become a standalone package. It has been ported to work on Linux, NetBSD, FreeBSD, OpenBSD, DragonFly BSD, Solaris, HP−UX and Cygwin.

SEE ALSO

sg_get_cpu_percents(3) **sg_get_disk_io_stats**(3) **sg_get_fs_stats**(3) **sg_get_host_info**(3) **sg_get_load_stats**(3) **sg_get_mem_stats**(3) **sg_get_network_io_stats**(3) **sg_get_network_iface_stats**(3) **sg_get_page_stats**(3) **sg_get_process_stats**(3) **sg_get_user_stats**(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_page_stats, sg_get_page_stats_diff − get paging statistics

SYNOPSIS

#include <statgrab.h>

sg_page_stats *_sg_get_page_stats_(void);

sg_page_stats *_sg_get_page_stats_diff_(void);

DESCRIPTION

sg_get_page_stats and sg_get_page_stats_diff both return a pointer to a static buffer of type sg_page_stats.

sg_get_page_stats will return the number of pages the system has paged in and out since bootup. sg_get_page_stats_diff will return the difference since last time it was called. If it has not been called before, it will return sg_get_page_stats.

RETURN VALUES

```
typedef struct{
    long long pages_pagein;
    long long pages_pageout;
    time_t systime;
}sg_page_stats;
```

pages_pagein
: The number of pages swapped into memory.

pages_pageout
: The number of pages swapped out of memory (to swap).

systime
: The time period over which pages_pagein and pages_pageout were transferred.

BUGS

Solaris doesn't seem to report accurately. It reports the number of pages swapped into memory, not necessarily from swap. This feature isn't deemed entirely reliable.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_cpu_stats, sg_get_cpu_stats_diff, sg_get_cpu_percents − get cpu usage

SYNOPSIS

#include <statgrab.h>

sg_cpu_percents *sg_get_cpu_percents(void);

sg_cpu_stats *sg_get_cpu_stats(void);

sg_cpu_stats *sg_get_cpu_stats_diff(void);

DESCRIPTION

sg_get_cpu_stats() and sg_get_cpu_stats_diff() both return static pointers of type sg_cpu_stats. sg_get_cpu_stats() returns the total amount of "ticks" the operating system has spent in each of the different states. sg_get_cpu_stats_diff() returns the difference in "ticks" for each of the states since last time sg_get_cpu_stats_diff() or sg_get_cpu_stats() was called. If it has never been called, it will return the result of sg_get_cpu_stats().

The value stored (the "ticks") will vary between operating systems. For example Solaris has a total of 100 per second, while Linux has substantially more. Also, different operating systems store different information − you won't find nice cpu on Solaris for example.

Because of this, you will ideally always want to work on a scale against the total, or in percentages.

sg_get_cpu_percents() returns a pointer to a static sg_cpu_percents. The function calls sg_get_cpu_stats_diff() and changes the values into percentages. If it has never been called before (and nor has sg_get_cpu_stats() or sg_get_cpu_stats_diff()), the returned percentages will be the systems total ever since its uptime. (Unless the counters have cycled)

RETURN VALUES

There are two structures returned by the CPU statistics functions.

```
typedef struct{
    long long user;
    long long kernel;
    long long idle;
    long long iowait;
    long long swap;
    long long nice;
    long long total;
    time_t systime;
}sg_cpu_stats;
```

```
typedef struct{
    float user;
    float kernel;
    float idle;
    float iowait;
    float swap;
    float nice;
    time_t time_taken;
}sg_cpu_percents;
```

user kernel idle iowait swap nice
: The different CPU states.

systime time_taken
The time taken in seconds since the last call of the function, or the system time.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_load_stats − get system load

SYNOPSIS

#include <statgrab.h>

sg_load_stats *sg_get_load_stats(void);

DESCRIPTION

This call returns a pointer to a static buffer of sg_load_stats.

On most systems this function is just a wrapper to the getloadavg system call.

RETURN VALUES

The structure returned is of type sg_load_stats.

```
typedef struct{
    double min1;
    double min5;
    double min15;
}sg_load_stats;
```

min1 The load average over 1 minute.

min5 The load average over 5 minutes.

min15 The load average over 15 minutes.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_cpu_stats, sg_get_cpu_stats_diff, sg_get_cpu_percents − get cpu usage

SYNOPSIS

#include <statgrab.h>

sg_cpu_percents *sg_get_cpu_percents(void);

sg_cpu_stats *sg_get_cpu_stats(void);

sg_cpu_stats *sg_get_cpu_stats_diff(void);

DESCRIPTION

sg_get_cpu_stats() and sg_get_cpu_stats_diff() both return static pointers of type sg_cpu_stats. sg_get_cpu_stats() returns the total amount of "ticks" the operating system has spent in each of the different states. sg_get_cpu_stats_diff() returns the difference in "ticks" for each of the states since last time sg_get_cpu_stats_diff() or sg_get_cpu_stats() was called. If it has never been called, it will return the result of sg_get_cpu_stats().

The value stored (the "ticks") will vary between operating systems. For example Solaris has a total of 100 per second, while Linux has substantially more. Also, different operating systems store different information − you won't find nice cpu on Solaris for example.

Because of this, you will ideally always want to work on a scale against the total, or in percentages.

sg_get_cpu_percents() returns a pointer to a static sg_cpu_percents. The function calls sg_get_cpu_stats_diff() and changes the values into percentages. If it has never been called before (and nor has sg_get_cpu_stats() or sg_get_cpu_stats_diff()), the returned percentages will be the systems total ever since its uptime. (Unless the counters have cycled)

RETURN VALUES

There are two structures returned by the CPU statistics functions.

```
typedef struct{
    long long user;
    long long kernel;
    long long idle;
    long long iowait;
    long long swap;
    long long nice;
    long long total;
    time_t systime;
}sg_cpu_stats;
```

```
typedef struct{
    float user;
    float kernel;
    float idle;
    float iowait;
    float swap;
    float nice;
    time_t time_taken;
}sg_cpu_percents;
```

user kernel idle iowait swap nice
: The different CPU states.

systime time_taken
The time taken in seconds since the last call of the function, or the system time.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_page_stats, sg_get_page_stats_diff − get paging statistics

SYNOPSIS

#include <statgrab.h>

sg_page_stats *_sg_get_page_stats_(void);

sg_page_stats *_sg_get_page_stats_diff_(void);

DESCRIPTION

sg_get_page_stats and sg_get_page_stats_diff both return a pointer to a static buffer of type sg_page_stats.

sg_get_page_stats will return the number of pages the system has paged in and out since bootup. sg_get_page_stats_diff will return the difference since last time it was called. If it has not been called before, it will return sg_get_page_stats.

RETURN VALUES

```
typedef struct{
    long long pages_pagein;
    long long pages_pageout;
    time_t systime;
}sg_page_stats;
```

pages_pagein
: The number of pages swapped into memory.

pages_pageout
: The number of pages swapped out of memory (to swap).

systime
: The time period over which pages_pagein and pages_pageout were transferred.

BUGS

Solaris doesn't seem to report accurately. It reports the number of pages swapped into memory, not necessarily from swap. This feature isn't deemed entirely reliable.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_disk_io_stats, sg_get_disk_io_stats_diff − get disk io statistics

SYNOPSIS

#include <statgrab.h>

sg_disk_io_stats *_sg_get_disk_io_stats_(int *_entries_);

sg_disk_io_stats *_sg_get_disk_io_stats_diff_(int *_entries_);

DESCRIPTION

Both calls take a pointer to an int, entries, which is filled with the number of disks the machine has. This is needed to know how many sg_disk_io_stats structures have been returned. A pointer is returned to the first sg_disk_io_stats.

sg_get_disk_io_stats returns the disk IO stored in the kernel which holds the amount of data transferred since bootup. On some platforms, such as Solaris 7, this value is stored in a 32bit int, so wraps around when it reaches 4GB. Other platforms, such as Solaris 8, hold the value in a 64bit int, which wraps somewhere near 17 million terabytes.

sg_get_disk_io_stats_diff is the same as sg_get_disk_io_stats except it will return the difference since the last call. So, for instance a call to sg_get_disk_io_stats_diff is made, and called again 5 seconds later. Over that time, 2000 bytes of traffic were written and 10000 bytes read. write_bytes will store 2000 bytes, read_bytes will store 10000, and systime will store 5. This function copes with wrap arounds by the O/S so should be seemless to use.

On Solaris libstatgrab will attempt to get the cXtXdXsX representation for the disk_name string. If it fails it will use a name like sd0. On some systems programs calling libstatgrab will need elevated privileges to lookup some of the names. The mappings are built up when sg_init is called.

RETURN VALUES

All diskio statistics return a pointer to a structure of type sg_disk_io_stats.

```
typedef struct{
    char *disk_name;
    long long read_bytes;
    long long write_bytes;
    time_t systime;
}sg_disk_io_stats;
```

disk_name
: The name known to the operating system. (eg. on linux it might be hda)

read_bytes
: The number of bytes the disk has read.

write_bytes
: The number of bytes the disk has written.

systime
: The time period over which read_bytes and write_bytes were transferred.

BUGS

On the very first call sg_get_disk_io_stats_diff will return the same as sg_get_disk_io_stats. After the first call it will always return the difference.

On operating systems that hold only 32bits of data there is a problem if the values wrap twice. For example, on Solaris 7 if 9GB is transferred and the operating system wraps at 4GB, the sg_get_disk_io_stats_diff function will return 5GB.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_network_io_stats, sg_get_network_io_stats_diff − get network statistics

SYNOPSIS

#include <statgrab.h>

sg_network_io_stats *sg_get_network_io_stats(int *entries);

sg_network_io_stats *sg_get_network_io_stats_diff(int *entries);

DESCRIPTION

Both calls take a pointer to an int, entries, which is filled with the number of network interfaces the machine has. This is needed to know how many sg_network_io_stats structures have been returned. A pointer is returned to the first sg_network_io_stats.

sg_get_network_io_stats returns the network traffic stored in the kernel which holds the amount of data transferred since bootup. On some platforms, such as Solaris 7, this value is stored in a 32bit int, so wraps around when it reaches 4GB. Other platforms, such as Solaris 8, hold the value in a 64bit int, which wraps somewhere near 17 million terabytes.

sg_get_network_io_stats also returns the number of packets sent and received, and the number of errors that have occured. It also makes the number of collisions available.

sg_get_network_io_stats_diff is the same as sg_get_network_io_stats except it will return the difference since the last call. So, for instance a call to sg_get_network_io_stats_diff is made, and called again 5 seconds later. Over that time, 20 bytes of traffic was transmitted and 10 bytes received. Tx will store 20, rx will store 10 and systime will store 5. This function copes with wrap arounds by the O/S so should be seemless to use.

RETURN VALUES

All network statistics return a pointer to a structure of type sg_network_io_stats.

```
typedef struct{
    char *interface_name;
    long long tx;
    long long rx;
    long long ipackets;
    long long opackets;
    long long ierrors;
    long long oerrors;
    long long collisions;
    time_t systime;
}sg_network_io_stats;
```

interface_name
: The name known to the operating system. (eg. on linux it might be eth0)

tx
: The number of bytes transmitted.

rx
: The number of bytes received.

ipackets
: The number of packets received.

opackets
: The number of packets transmitted.

ierrors
: The number of receive errors.

oerrors
: The number of transmit errors.

collisions
The number of collisions.

systime The time period over which tx and rx were transferred.

BUGS

On the very first call sg_get_network_io_stats_diff will return the same as sg_get_network_io_stats. After the first call it will always return the difference.

On operating system that hold only 32bits of data there is a problem if the values wrap twice. For example, on Solaris 7 if 9GB is transferred and the operating system wraps at 4GB, the sg_get_network_io_stats_diff function will return 5GB.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_network_io_stats, sg_get_network_io_stats_diff − get network statistics

SYNOPSIS

#include <statgrab.h>

sg_network_io_stats *sg_get_network_io_stats(int *entries);

sg_network_io_stats *sg_get_network_io_stats_diff(int *entries);

DESCRIPTION

Both calls take a pointer to an int, entries, which is filled with the number of network interfaces the machine has. This is needed to know how many sg_network_io_stats structures have been returned. A pointer is returned to the first sg_network_io_stats.

sg_get_network_io_stats returns the network traffic stored in the kernel which holds the amount of data transferred since bootup. On some platforms, such as Solaris 7, this value is stored in a 32bit int, so wraps around when it reaches 4GB. Other platforms, such as Solaris 8, hold the value in a 64bit int, which wraps somewhere near 17 million terabytes.

sg_get_network_io_stats also returns the number of packets sent and received, and the number of errors that have occured. It also makes the number of collisions available.

sg_get_network_io_stats_diff is the same as sg_get_network_io_stats except it will return the difference since the last call. So, for instance a call to sg_get_network_io_stats_diff is made, and called again 5 seconds later. Over that time, 20 bytes of traffic was transmitted and 10 bytes received. Tx will store 20, rx will store 10 and systime will store 5. This function copes with wrap arounds by the O/S so should be seemless to use.

RETURN VALUES

All network statistics return a pointer to a structure of type sg_network_io_stats.

```
typedef struct{
    char *interface_name;
    long long tx;
    long long rx;
    long long ipackets;
    long long opackets;
    long long ierrors;
    long long oerrors;
    long long collisions;
    time_t systime;
}sg_network_io_stats;
```

interface_name
: The name known to the operating system. (eg. on linux it might be eth0)

tx
: The number of bytes transmitted.

rx
: The number of bytes received.

ipackets
: The number of packets received.

opackets
: The number of packets transmitted.

ierrors
: The number of receive errors.

oerrors
: The number of transmit errors.

collisions
The number of collisions.

systime The time period over which tx and rx were transferred.

BUGS

On the very first call sg_get_network_io_stats_diff will return the same as sg_get_network_io_stats. After the first call it will always return the difference.

On operating system that hold only 32bits of data there is a problem if the values wrap twice. For example, on Solaris 7 if 9GB is transferred and the operating system wraps at 4GB, the sg_get_network_io_stats_diff function will return 5GB.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

statgrab−make−mrtg−index − generates MRTG configuration

SYNOPSIS

statgrab−make−mrtg−index [OPTION]... [CONFIGFILE]...

DESCRIPTION

statgrab−make−mrtg−index generates an XHTML index page from MRTG configuration files specified on the command line or read from stdin.

OPTIONS

The following options are supported.

−−title *TITLE*
: Use TITLE as the title of the generated page

−−help Display help and exit

SEE ALSO

statgrab(1) **statgrab−make−mrtg−config**(1)

AUTHORS

This man page was derived from the man page written by Bartosz Fenski for the Debian GNU/Linux distribution.

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_process_stats, sg_get_process_count, sg_process_compare_name, sg_process_compare_pid, sg_process_compare_uid, sg_process_compare_gid, sg_process_compare_size, sg_process_compare_res, sg_process_compare_cpu, sg_process_compare_time − get process statistics

SYNOPSIS

#include <statgrab.h>

sg_process_stats *sg_get_process_stats(int *entries);

sg_process_count *sg_get_process_count(void);

int sg_process_compare_name(const void *va, const void *vb);

int sg_process_compare_pid(const void *va, const void *vb);

int sg_process_compare_uid(const void *va, const void *vb);

int sg_process_compare_gid(const void *va, const void *vb);

int sg_process_compare_size(const void *va, const void *vb);

int sg_process_compare_res(const void *va, const void *vb);

int sg_process_compare_cpu(const void *va, const void *vb);

int sg_process_compare_time(const void *va, const void *vb);

DESCRIPTION

The sg_get_process_stats takes a pointer to an int, entries, which is filled with the number of processes the snapshot contains. The return value is a pointer to the first member of an array of sg_process_stats structures; the number of entries in the array is returned in entries.

sg_get_process_count returns a pointer to a static buffer of type sg_process_count.

These two functions provide information on the process running on the system. In the first case lots of detail is provided, and in the second case a summary of the states of the current processes.

sg_process_compare_name

sg_process_compare_pid

sg_process_compare_uid

sg_process_compare_gid

sg_process_compare_size

sg_process_compare_res

sg_process_compare_cpu

sg_process_compare_time

These functions compare two sg_process_stats entries, and return an int to represent which one is greater. The main use of these functions is to be passed to qsort to sort the sg_process_stats by the given type.

For example:

qsort(sg_process_stats ps, int ps_size, sizeof *ps, sg_process_compare_pid);

RETURN VALUES

The structure returned by sg_get_process_stats is of type sg_process_stats.

```
typedef struct {
    char *process_name;
    char *proctitle;
```

```
    pid_t pid;
    pid_t parent; /* Parent pid */
    pid_t pgid;   /* process id of process group leader */

    uid_t uid;
    uid_t euid;
    gid_t gid;
    gid_t egid;

    unsigned long long proc_size; /* in bytes */
    unsigned long long proc_resident; /* in bytes */
    time_t time_spent; /* time running in seconds */
    double cpu_percent;
    int nice;
    sg_process_state state;
} sg_process_stats;


typedef enum {
    SG_PROCESS_STATE_RUNNING,
    SG_PROCESS_STATE_SLEEPING,
    SG_PROCESS_STATE_STOPPED,
    SG_PROCESS_STATE_ZOMBIE,
    SG_PROCESS_STATE_UNKNOWN
} sg_process_state;
```

process_name
: The name of the command that was run.

proctitle
: The command line (the "title") of the process. Take note − this can be modifi ed by the process, so isn’t guaranteed to be the original command line.

pid
: The process ID.

parent
: The parent process ID.

pgid
: The process ID of the process group leader.

uid
: The UID the process is running as.

euid
: The effective UID the process is running as.

gid
: The GID the process is running as.

egid
: The effective GID the process is running as.

proc_size
: The size of the process in bytes.

proc_resident
: The size of the process that’s resident in memory.

time_spent
: The number of seconds the process has been running.

cpu_percent
: The current percentage of CPU the process is using.

nice
: The nice value of the process.

state The current state of the process. See sg_process_state for permitted values.

The structure returned by sg_get_process_count is of type sg_process_count.

```
typedef struct{
        int total;
        int running;
        int sleeping;
        int stopped;
        int zombie;
}sg_process_count;
```

total The total number of processes.

running
The number of running processes.

sleeping
The number of sleeping processes.

stopped
The number of stopped processes.

zombie The number of zombie processes.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_host_info − get general operating system statistics

SYNOPSIS

#include <statgrab.h>

sg_host_info *sg_get_host_info(void);

DESCRIPTION

This call returns details on the operating system.

It returns a pointer to a static buffer of sg_host_info.

RETURN VALUES

The structure returned is of type sg_host_info.

```
typedef struct{
    char *os_name;
    char *os_release;
    char *os_version;
    char *platform;
    char *hostname;
    time_t uptime;
}sg_host_info;
```

os_name
: the operating system name. (eg. SunOS or Linux)

os_release
: the operating system release. (eg. 5.8 or 5.9 or Solaris)

os_version
: the version level of the OS.

platform
: the hardware platform (architecture) the OS runs on.

hostname
: the name of the machine.

uptime
: the uptime of the machine in seconds.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_mem_stats, sg_get_swap_stats − get VM statistics

SYNOPSIS

#include <statgrab.h>

sg_mem_stats *sg_get_mem_stats(void);

sg_swap_stats *sg_get_swap_stats(void);

DESCRIPTION

Memory statistics are accessed through the sg_get_mem_stats function. It returns a pointer to a static sg_mem_stats.

The sg_get_swap_stats returns returns swap statistics. It returns a pointer to a static sg_swap_stats.

On the FreeBSD operating system elevated privileges are required to access the swap statistics. Making the program setgid kmem should be sufficient. Programs running as root will not have this problem.

RETURN VALUES

The VM system calls can return a pointer to either a sg_mem_stats or a sg_swap_stats.

```
typedef struct{
    long long total;
    long long free;
    long long used;
    long long cache;
}sg_mem_stats;
```

total	The total amount of memory in bytes.
free	The total free memory in bytes.
used	The total used memory in bytes.
cache	The amount of cache used in bytes.

```
typedef struct{
    long long total;
    long long used;
    long long free;
}sg_swap_stats;
```

total	The total swap space in bytes.
used	The used swap in bytes.
free	The free swap in bytes.

TODO

Add a function to hold open the file descriptor to the kernel memory structures. Doing this would allow the elevated privileges to be dropped early on.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

saidar − a curses−based tool for viewing system statistics

SYNOPSIS

saidar [−d **delay**] [−c] [−v] [−h]

DESCRIPTION

saidar is a curses−based tool for viewing the system statistics available through libstatgrab. Statistics include CPU, processes, load, memory, swap, network I/O, disk I/O, and file system information.

OPTIONS

The following options are supported.

−d *DELAY*
: Wait DELAY seconds between updates (default 3)

−c
: Enables coloured output

−v
: Prints the version number

−h
: Display help and exits

COLOURED OUTPUT

When the **−c** option is used saidar uses colours to display the data. Each area has a different colour to distinguish it from the nearby fields. Bold and reverse video effects are used to emphasis fields where necessary.

The load average figures are displayed in bold when they're notably (greater that 1) different to each other. This purely shows that a change in the load is occuring.

When CPU usage goes over 60% it will be displayed in bold. At 90% the field is displayed using reverse video. Likewise for memory, swap and disk usage at 75% and 90%. Zombie processes are also highlighted.

Other values (paging, disk I/O and network I/O) are not highlighted due to the nature of the values; it's not easy to determine what thresholds are significant.

SEE ALSO

statgrab(1) **statgrab**(3)

AUTHORS

This man page was derived from the man page written by Bartosz Fenski for the Debian GNU/Linux distribution.

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_disk_io_stats, sg_get_disk_io_stats_diff − get disk io statistics

SYNOPSIS

#include <statgrab.h>

sg_disk_io_stats *_sg_get_disk_io_stats_(int *_entries_);

sg_disk_io_stats *_sg_get_disk_io_stats_diff_(int *_entries_);

DESCRIPTION

Both calls take a pointer to an int, entries, which is filled with the number of disks the machine has. This is needed to know how many sg_disk_io_stats structures have been returned. A pointer is returned to the first sg_disk_io_stats.

sg_get_disk_io_stats returns the disk IO stored in the kernel which holds the amount of data transferred since bootup. On some platforms, such as Solaris 7, this value is stored in a 32bit int, so wraps around when it reaches 4GB. Other platforms, such as Solaris 8, hold the value in a 64bit int, which wraps somewhere near 17 million terabytes.

sg_get_disk_io_stats_diff is the same as sg_get_disk_io_stats except it will return the difference since the last call. So, for instance a call to sg_get_disk_io_stats_diff is made, and called again 5 seconds later. Over that time, 2000 bytes of traffic were written and 10000 bytes read. write_bytes will store 2000 bytes, read_bytes will store 10000, and systime will store 5. This function copes with wrap arounds by the O/S so should be seemless to use.

On Solaris libstatgrab will attempt to get the cXtXdXsX representation for the disk_name string. If it fails it will use a name like sd0. On some systems programs calling libstatgrab will need elevated privileges to lookup some of the names. The mappings are built up when sg_init is called.

RETURN VALUES

All diskio statistics return a pointer to a structure of type sg_disk_io_stats.

```
typedef struct{
    char *disk_name;
    long long read_bytes;
    long long write_bytes;
    time_t systime;
}sg_disk_io_stats;
```

disk_name
: The name known to the operating system. (eg. on linux it might be hda)

read_bytes
: The number of bytes the disk has read.

write_bytes
: The number of bytes the disk has written.

systime
: The time period over which read_bytes and write_bytes were transferred.

BUGS

On the very first call sg_get_disk_io_stats_diff will return the same as sg_get_disk_io_stats. After the first call it will always return the difference.

On operating systems that hold only 32bits of data there is a problem if the values wrap twice. For example, on Solaris 7 if 9GB is transferred and the operating system wraps at 4GB, the sg_get_disk_io_stats_diff function will return 5GB.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

sg_get_user_stats − get the current logged in users

SYNOPSIS

#include <statgrab.h>

sg_user_stats *_sg_get_user_stats_(void);

DESCRIPTION

This call returns a pointer to a static buffer of sg_user_stats.

RETURN VALUES

The structure returned is of type sg_user_stats.

```
typedef struct{
	char *name_list;
	int num_entries;
}sg_user_stats;
```

name_list

A pointer to a space seperated list of the currently logged in users.

num_entries

The number of users currently logged in.

SEE ALSO

statgrab(3)

WEBSITE

http://www.i−scream.org/libstatgrab/

NAME

pktstat − display packet activity on a crt

SYNOPSIS

pktstat [**−1BcFlnpPtT**] [**−a** *abbrev*] [**−A** *file*] [**−i** *interface*] [**−k** *keeptime*] [**−m** *maxbps*] [**−w** *waittime*] [*filter-expr*]

DESCRIPTION

The **pktstat** program displays a real-time summary of packet activity on an interface. Each line displays the data rate associated with different classes of packets.

pktstat understands the following command line options:

−1 Single-shot (batch) mode. **pktstat** collects data for *waittime* seconds (see **−w** option) then emits a line indicating the number of flows detected, and the period of data capture in seconds. Then, each flow line is printed in the form of the number of data link octets associated with the flow, the number of data link frames (packets), and then the flow description.

−a *abbrev*
Add *abbrev* to the list of abbreviation patterns. (See below for details.)

−A *file*
Read abbreviation patterns from the given *file*. (See **Abbreviations**, below.) If the option **−A none** is given, then default abbreviation files are not loaded.

−B Display data rates in bytes per second (Bps) instead of in bits per second (bps).

−c Do not combine some packet classes into one class. For example, TCP connections are kept as two separate flows.

−F Show full hostnames. Normally, hostnames are truncated to the first component of their domain name before display.

−i *interface*
Listen on the given interface. If not specified, a suitable interface is chosen.

−k *keeptime*
When no packets have been seen for a particular class, retain an entry on the display for this many screen seconds. Defaults to 10.

−l Display and sort flows by when they were last seen. (Incompatible with **−t**)

−m *maxbps*
Fix the maximum bit rate for the interface at *maxbps* instead of auto-detecting it.

−n Do not try and resolve hostnames or service port numbers.

−p Show packet counts instead of bit counts.

−P Do not try to put the interface into promiscuous mode.

−t "Top" mode. Sorts the display by bit count (or packet count if **−p** was given) instead of by the name.

−T Show totals.

−w *waittime*
Refresh the display every *waittime* seconds. The default is 5 seconds.

filter-expr

Only consider packets matching the given *filter-expr*. If no filter is provided, all packets are considered. See `tcpdump`(8) for information on valid expressions.

If the terminal supports it, the display briefly highlights in bold new connections or old connections carrying data after a period of inactivity.

Simple statistics about the interface are also displayed such as the current and average bit rates (measured just above the data link layer). Load averages refer to bit rate decayed averages for the last 1, 5 and 15 minutes.

During display, the following keystrokes are recognised:

q	quit
Ctrl-L	redraw screen
t	toggle the **-t** flag (top mode)
T	toggle the **-T** flag (totals mode)
w	allows changing of the **-w** flag value (wait time)
n	toggle the **-n** flag (numeric display)
p	toggle the **-p** flag (packets instead of bits)
b | **B**	toggle the **-B** flag (bps or Bps)
f | **F**	toggle the **-F** flag (full hostnames)
r	reset collected statistics (min, max, etc.), flush flow history and reset DNS/service and fragment caches
l	show and sort flows by when they were last active
?	toggle display of help/status text at the bottom of the display

Packet classes

All packet classes, or flows, are "tagged" with a descriptive string, such as `tcp ftpserver:20524 <-> cathexis:17771`.

In addition to being tagged, some protocol-state information can be associated with a flow. This is displayed immediately below a flow line. Descriptive information for FTP, HTTP, X11 and SUP connections is determined from simple decoding of some packets. If the connection is 'open', it is introduced with a right angle shape (+), otherwise it is introduced with a hyphen character.

```
tcp www:80 <-> hamartia:19179
+ GET /index.html
```

Abbreviations

Abbreviation patterns are a way of further combining flows. As packets are decoded, their flow name is constructed at the various protocol layers. At address combining stage (where arrows such as '`->`' are inserted) and at the final display stage, flow names are checked against a list of abbreviation patterns, and the abbreviation's name substituted if a match is found. For example, the pattern `* <-> *:domain` will match DNS packets in both the UDP and TCP layers.

Abbreviations take the form [*abbrev*@]*pattern*. The *pattern* part can contain the wildcard character, asterisk '*' which matches zero or more non-space characters. The space character matches one or more whitespace characters. Leading and trailing spaces are ignored.

If the optional *abbrev* is not specified, the the pattern text itself is used as the abbreviation.

Patterns are checked in the order given on the command line or in the files, i.e. as soon as one of the patterns matches a tag, no further patterns are considered. Recall that patterns can be applied multiple times to a tag.

A patterns file can contain blank lines, which are ignored. Comment lines that commence with a '#' character are also ignored.

After processing all command line abbreviations and abbreviation files, **pktstat** looks for and loads the files `.pktstatrc`, `$HOME/.pktstatrc` and `/var/opt/UNItools/etc/pktstatrc`. This behaviour is suppressed by supplying an **−A none** option.

EXAMPLES

Here are the contents of my `.pktstatrc` file:

```
dns @ udp *:domain <-> *
dns @ udp * <-> *:domain
irc @ udp 192.168.0.81:6666 <-> *
```

SEE ALSO

`bpf`(4), `tcpdump`(8)

AUTHORS

David Leonard, `leonard@users.sourceforge.net`

BUGS

DNS lookups can take too much time, possibly leading to missed packets.

The data rates do not take into account data link framing overhead or compression savings at the data link layer.

The direction of traffic is not taken into account: both ingress and egress data rates are combined. If you want to separate them, you will need to use a filter expression.

Descriptive information for X11, FTP, HTTP and SUP flows is derived from the very first packets sent on those protocols. If you start **pktstat** after any of these flows have commenced, there may be no description available for them.

NAME

minicom − friendly serial communication program

SYNOPSIS

minicom [-somMlwz8] [-c on|off] [-S script] [-d entry]
[-a on|off] [-t term] [-p pty] [-C capturefile] [*configuration*]

DESCRIPTION

minicom is a communication program which somewhat resembles the shareware program TELIX but is free with source code and runs under most unices. Features include dialing directory with auto-redial, support for UUCP-style lock files on serial devices, a seperate script language interpreter, capture to file, multiple users with individual configurations, and more.

COMMAND-LINE

-s **S**etup. Root edits the system-wide defaults in /etc/minirc.dfl with this option. When it is used, minicom does *not* initialize, but puts you directly into the configuration menu. This is very handy if minicom refuses to start up because your system has changed, or for the first time you run minicom. For most systems, reasonable defaults are already compiled in.

-o Do not initialize. Minicom will skip the initialization code. This option is handy if you quitted from minicom without resetting, and then want to restart a session. It is potentially dangerous though: no check for lock files etc. is made, so a normal user could interfere with things like uucp... Maybe this will be taken out later. For now it is assumed, that users who are given access to a modem are responsible enough for their actions.

-m Override command-key with the Meta or ALT key. This is the default in 1.80 and it can also be configured in one of minicom's menus, but if you use different terminals all the time, of which some don't have a Meta or ALT key, it's handy to set the default command key to Ctrl-A and use this option when you have a keyboard supporting Meta or ALT keys. Minicom assumes that your Meta key sends the ESC prefix, not the other variant that sets the highest bit of the character.

-M Same as -m, but assumes that your Meta key sets the 8th bit of the character high (sends 128 + character code).

-z Use terminal status line. This only works on terminals that support it and that have the relevant information in their *termcap* or *terminfo* database entry.

-l **L**iteral translation of characters with the high bit set. With this flag on, minicom will not try to translate the IBM line characters to ASCII, but passes them straight trough. Many PC-unix clones will display them correctly without translation (Linux in a special mode, Coherent and Sco).

-L Ditto but assume screen uses an ISO8859 character set.

-w Turns linewrap on at startup by default.

-a **A**ttribute usage. Some terminals, notably televideo's, have a rotten attribute handling (serial instead of parallel). By default, minicom uses '-a on', but if you are using such a terminal you can (must!) supply the option '-a off'. The trailing 'on' or 'off' is needed.

-t **T**erminal type. With this flag, you can override the environment TERM variable. This is handy for use in the MINICOM environment variable; one can create a special termcap entry for use with minicom on the console, that initializes the screen to raw mode so that in conjunction with the -l flag, the IBM line characters are displayed untranslated.

-c **C**olor usage. Some terminals (such as the Linux console) support color with the standard ANSI escape sequences. Because there is apparently no termcap support for color, these escape sequences are hard-coded into minicom. Therefore this option is off by default. You can turn it on with '-c on'. This, and the '-m' option, are good candidates to put into the MINICOM environment variable.

-S **script**. Run the named script at startup. So far, passing username and password to a startup script is not supported. If you also use the -d option to start dialing at startup, the -S script will be run BEFORE dialing the entries specified with -d.

-d **D**ial an entry from the dialing directory on startup. You can specify an index number, but also a substring of the name of the entry. If you specify a name that has multiple entries in the directory, they are all tagged for dialing. You can also specify multiple names or index numbers by separating them with commas. The dialing will start from the first entry specified after all other program initialization procedures are completed.

-p **P**seudo terminal to use. This overrrides the terminal port defined in the configuration files, but only if it is a pseudo tty. The filename supplied must be of the form (/dev/)tty[p-z/][0-f], (/dev/)pts[p-z/][0-f] or (/dev/)pty[p-z/][0-f]. For example, /dev/ttyp1, pts/0 or /dev/ptyp2.

-C **filename**. Open capture file at startup.

-T Disable the display of the online time in the status bar.

-8 8bit characters pass through without any modification. 'Continuous' means no locate/attribute control sequences are inserted without real change of locate/attribute. This mode is to display 8bit multibyte characters such as Japanese. Not needed in every language with 8bit characters. (For example displaying Finnish text doesn't need this.)

When **minicom** starts, it first searches the MINICOM environment variable for command-line arguments, which can be over-ridden on the command line. Thus, if you have done

```
MINICOM='-m -c on'
export MINICOM
```

or the equivalent, and start minicom, minicom will assume that your terminal *has* a Meta or <ALT> key and that color is supported. If you then log in from a terminal without color support, and you have set MINICOM in your startup (.profile or equivalent) file, and don't want to re-set your environment variable, you can type 'minicom -c off' and run without color support for that session.

configuration

The *configuration* argument is more interesting. Normally, minicom gets its defaults from a file called "minirc.dfl". If you however give an argument to minicom, it will try to get its defaults from a file called "minirc.*configuration*". So it is possible to create multiple configuration files, for different ports, different users etc. Most sensible is to use device names, such as tty1, tty64, sio2 etc. If a user creates his own configuration file, it will show up in his home directory as '.minirc.dfl'.

USE

Minicom is window based. To popup a window with the function you want, press Control-A (from now on, we will use C-A to mean Control-A), and then the function key (a-z or A-Z). By pressing C-A first and then 'z', a help screen comes up with a short summary of all commands. This escape key can be altered when minicom is configured (-s option or C-A O), but we'll stick to Control-A for now.

For every menu the next keys can be used:

UP	arrow-up or 'k'
DOWN	arrow-down or 'j'
LEFT	arrow-left or 'h'
RIGHT	arrow-right or 'l'
CHOOSE	Enter
CANCEL	ESCape.

The screen is divided into two portions: the upper 24 lines are the terminal-emulator screen. In this window, ANSI or VT100 escape sequences are interpreted. If there is a line left at the bottom, a status line is placed there. If this is not possible the status line will be showed every time you press C-A. On terminals that have a special status line that will be used if the termcap information is complete *and* the **-k** flag has been given.

Possible commands are listed next, in alphabetical order.

C-A Pressing C-A a second time will just send a C-A to the remote system. If you have changed your "escape character" to something other than C-A, this works analogously for that character.

A Toggle 'Add Linefeed' on/off. If it is on, a linefeed is added before every carriage return displayed on the screen.

B Gives you a scroll back buffer. You can scroll up with **u**, down with **d**, a page up with **b**, a page down with **f**, and if you have them the **arrow** and **page up/page down** keys can also be used. You can search for text in the buffer with **s** (case-sensitive) or **S** (case-insensitive). **N** will find the next occurrence of the string. **c** will enter citation mode. A text cursor appears and you specify the start line by hitting Enter key. Then scroll back mode will finish and the contents with prefix '>' will be sent.

C Clears the screen.

D Dial a number, or go to the dialing directory.

E Toggle local echo on and off (if your version of minicom supports it).

F A break signal is sent to the modem.

G Run script (Go). Runs a login script.

H Hangup.

I Toggle the type of escape sequence that the cursor keys send between normal and applications mode. (See also the comment about the status line below).

J Jump to a shell. On return, the whole screen will be redrawn.

K Clears the screen, runs kermit and redraws the screen upon return.

L Turn Capture file on off. If turned on, all output sent to the screen will be captured in the file too.

M Sends the modem initialization string. If you are online and the DCD line setting is on, you are asked for confirmation before the modem is initialized.

O Configure minicom. Puts you in the configuration menu.

P Communication Parameters. Allows you to change the bps rate, parity and number of bits.

Q Exit minicom without resetting the modem. If macros changed and were not saved, you will have a chance to do so.

R Receive files. Choose from various protocols (external). If you have the filename selection window and the prompt for download directory enabled, you'll get a selection window for choosing the directory for downloading. Otherwise the download directory defined in the Filenames and paths menu will be used.

S Send files. Choose the protocol like you do with the receive command. If you don't have the filename selection window enabled (in the File transfer protocols menu), you'll just have to write the filename(s) in a dialog window. If you have the selection window enabled, a window will pop up showing the filenames in your upload directory. You can tag and untag filenames by pressing spacebar, and move the cursor up and down with the cursor keys or j/k. The selected filenames are shown highlighted. Directory names are shown [within brackets] and you can move up or down in the directory tree by pressing the spacebar twice. Finally, send the files by pressing ENTER or quit by pressing ESC.

T Choose Terminal emulation: Ansi(color) or vt100. You can also change the backspace key here, turn the status line on or off, and define delay (in milliseconds) after each newline if you need that.

W Toggle linewrap on/off.

X Exit minicom, reset modem. If macros changed and were not saved, you will have a chance to do so.

Z Pop up the help screen.

DIALING DIRECTORY

By pressing C-A D the program puts you in the dialing directory. Select a command by pressing the capitalized letter or moving cursor right/left with the arrow keys or the h/l keys and pressing Enter. You can add, delete or edit entries and move them up and down in the directory list. By choosing "dial" the phone numbers of the tagged entries, or if nothing is tagged, the number of the highlighted entry will be dialed. While the modem is dialing, you can press escape to cancel dialing. Any other key will close the dial window, but won't cancel the dialing itself. Your dialing directory will be saved into a the file ".dialdir" in your home directory. You can scroll up and down with the arrow keys, but you can also scroll complete pages by

pressing the PageUp or PageDown key. If you don't have those, use Control-B (Backward) and Control-F (Forward). You can use the space bar to **tag** a number of entries and minicom will rotate trough this list if a connection can't be made. A '>' symbol is drawn in the directory before the names of the tagged entries.

The "edit" menu speaks for itself, but I will discuss it briefly here.

A - Name The name for this entry

B - Number and its telephone number.

C - Dial string #
Which specific dial string you want to use to connect. There are three different dial strings (prefixes and suffixes) that can be configured in the **Modem and dialing** menu.

D - Local echo can be on or off for this system (if your version of minicom supports it).

E - Script The script that must be executed after a succesfull connection is made (see the manual for runscript)

F - Username The username that is passed to the runscript program. It is passed in the environment string "$LOGIN".

G - Password The password is passed as "$PASS".

H - Terminal Emulation
Use ANSI or VT100 emulation.

I - Backspace key sends
What code (Backspace or Delete) the backspace key sends.

J - Linewrap Can be on or off.

K - Line settings
Bps rate, bits, parity and number of stop bits to use for this connection. You can choose **current** for the speed, so that it will use whatever speed is being used at that moment (useful if you have multiple modems).

L - Conversion table
You may spacify a character conversion table to be loaded whenever this entry answers, before running the login script. If this field is blank, the conversion table stays unchanged.

The edit menu also shows the latest date and time when you called this entry and the total number of calls there, but doesn't let you change them. They are updated automatically when you connect.

The moVe command lets you move the highlighted entry up or down in the dialing directory with the up/down arrow keys or the k and j keys. Press Enter or ESC to end moving the entry.

CONFIGURATION

By pressing C-A O you will be thrown into the setup menu. Most settings there can be changed by everyone, but some are restricted to root only. Those priviliged settings are marked with a star (*) here.

Filenames and paths

This menu defines your default directories.

A - Download directory
where the downloaded files go to.

B - Upload directory
where the uploaded files are read from.

C - Script directory
Where you keep your login scripts.

D - Script program
Which program to use as the script interpreter. Defaults to the program "runscript", but if you want to use something else (eg, /bin/sh or "expect") it is possible. Stdin and stdout are connected to the modem, stderr to the screen.
If the path is relative (ie, does not start with a slash) then it's relative to your home directory, except for the script interpreter.

E - Kermit program

Where to find the executable for kermit, and it's options. Some simple macro's can be used on the command line: '%l' is expanded to the complete filename of the dial out-device, '%f' is expanded to the serial port file descriptor and '%b' is expanded to the current serial port speed.

F - Logging options

Options to configure the logfile writing.

A - File name

Here you can enter the name of the logfile. The file will be written in your home directory, and the default value is "minicom.log". If you blank the name, all logging is turned off.

B - Log connects and hangups

This option defines whether or not the logfile is written when the remote end answers the call or hangs up. Or when you give the hangup command yourself or leave minicom without hangup while online.

C - Log file transfers

Do you want log entries of receiving and sending files.

The 'log' command in the scripts is not affected by logging options B and C. It is always executed, if you just have the name of the log file defined.

File Transfer Protocols

Protocols defined here will show up when C-A s/r is pressed. "Name" in the beginning of the line is the name that will show up in the menu. "Program" is the path to the protocol. "Name" after that defines if the program needs an argument, eg. a file to be transmitted. U/D defines if this entry should show up in the upload or the download menu. Fullscr defines if the program should run full screen, or that minicom will only show it's stderr in a window. IO-Red defines if minicom should attach the program's standard in and output to the modem port or not. "Multi" tells the filename selection window whether or not the protocol can send multiple files with one command. It has no effect on download protocols, and it is also ignored with upload protocols if you don't use the filename selection window. The old sz and rz are not full screen, and have IO-Red set. However, there are curses based versions of at least rz that do not want their stdin and stdout redirected, and run full screen. All file transfer protocols are run with the UID of the user, and not with UID=root. '%l', '%f' and '%b' can be used on the command line as with kermit. Within this menu you can also define if you want to use the filename selection window when prompted for files to upload, and if you like to be prompted for the download directory every time the automatic download is started. If you leave the download directory prompt disabled, the download directory defined in the file and directory menu is used.

Serial port setup

***A - Serial device**

/dev/tty1 or /dev/ttyS1 for most people. /dev/cua<n> is still possible under linux, but not recommended any more because these devices are obsolete and many newly installed systems with kernel 2.2.x or newer don't have them. Use /dev/ttyS<n> instead. You may also have /dev/modem as a symlink to the real device.

If you have modems connected to two or more serial ports, you may specify all of them here in a list separated by space, comma or semicolon. When Minicom starts, it checks the list until it finds an available modem and uses that one. (However, you can't specify different init strings to them ..at least not yet.)

To use a UNIX socket for communication the device name must be prefixed with "unix#" following by the full path and the filename of the socket. Minicom will then try to connect to this socket as a client. As long as it cannot connect to the socket it stays 'offline'. As soon as the connection establishes, minicom goes 'online'. If the server closes the socket, minicom

switches to 'offline' again.

***B - Lock file location**

On most systems This should be /usr/spool/uucp. Linux systems use /var/lock. If this directory does not exist, minicom will not attempt to use lockfiles.

***C - Callin program**

If you have a uugetty or something on your serial port, it could be that you want a program to be run to switch the modem cq. port into dialin/dialout mode. This is the program to get into dialin mode.

***D - Callout program**

And this to get into dialout mode.

E - Bps/Par/Bits

Default parameters at startup.

If one of the entries is left blank, it will not be used. So if you don't care about locking, and don't have a getty running on your modemline, entries B - D should be left blank. Be warned! The callin and callout programs are run with the effective user id of "root", eg 0!

Modem and Dialing

Here, the parameters for your modem are defined. I will not explain this further because the defaults are for generic Hayes modems, and should work always. This file is not a Hayes tutorial :-) The only things worth noticing are that control characters can be sent by prefixing them with a '^', in which '^^' means '^' itself, and the '\' character must also be doubled as '\\', because backslash is used specially in the macro definitions. Some options however, don't have much to do with the modem but more with the behaviour of minicom itself:

M - Dial time

The number of seconds before minicom times out if no connection is established.

N - Delay before redial

Minicom will redial if no connection was made, but it first waits some time.

O - Number of tries

Maximum number of times that minicom attempts to dial.

P - Drop DTR time

If you set this to 0, minicom hangs up by sending a Hayes-type hangup sequence. If you specify a non-zero value, the hangup will be done by dropping the DTR line. The value tells in seconds how long DTR will be kept down.

Q - Auto bps detect

If this is on, minicom tries to match the dialed party's speed. With most modern modems this is NOT desirable, since the modem buffers the data and converts the speed.

R - Modem has DCD line

If your modem, and your O/S both support the DCD line (that goes 'high' when a connection is made) minicom will use it. When you have this option on, minicom will also NOT start dialing while you are already online.

S - Status line shows DTE speed / line speed

You can toggle the status line to show either the DTE speed (the speed which minicom uses to communicate with your modem) or the line speed (the speed that your modem uses on the line to communicate with the other modem). Notice that the line speed may change during the connection, but you will still only see the initial speed that the modems started the connection with. This is because the modem doesn't tell the program if the speed is changed. Also, to see the line speed, you need to have the modem set to show it in the connect string. Otherwise you will only see 0 as the line speed.

T - Multi-line untag

You can toggle the feature to untag entries from the dialing directory when a connection is established to a multi-line BBS. All the tagged entries that have the same name are untagged.

Note that a special exception is made for this menu: every user can change all parameters

here, but some of them will not be saved.

Screen and keyboard

A - Command key is

the 'Hot Key' that brings you into command mode. If this is set to 'ALT' or 'meta key', you can directly call commands by alt-key instead of HotKey-key.

B - Backspace key sends

There still are some systems that want a VT100 to send DEL instead of BS. With this option you can enable that stupidity. (Eh, it's even on by default...)

C - Status line is

Enabled or disabled. Some slow terminals (for example, X-terminals) cause the status line to jump "up and down" when scrolling, so you can turn it off if desired. It will still be shown in command-mode.

D - Alarm sound

If turned on, minicom will sound an alarm (on the console only) after a succesfull connection and when up/downloading is complete.

E - Foreground Color (menu)

indicates the foreground color to use for all the configuration windows in minicom.

F - Background Color (menu)

indicates the background color to use for all the configuration windows in minicom. Note that minicom will not allow you to set forground and background colors to the same value.

G - Foreground Color (term)

indicates the foreground color to use in the terminal window.

H - Background Color (term)

indicates the background color to use in the terminal window. Note that minicom will not allow you to set forground and background colors to the same value.

I - Foreground Color (stat)

indicates the foreground color to use in for the status bar.

J - Background Color (stat)

indicates the color to use in for the status bar. Note that minicom will allow you to set the status bar's forground and background colors to the same value. This will effectively make the status bar invisible but if these are your intensions, please see the option

K - History buffer size

The number of lines to keep in the history buffer (for backscrolling).

L - Macros file

is the full path to the file that holds macros. Macros allow you to define a string to be sent when you press a certain key. In minicom, you may define F1 through F10 to send up to 256 characters [this is set at compile time]. The filename you specify is verified as soon as you hit ENTER. If you do not have permissions to create the specified file, an error message will so indicate and you will be forced to re-edit the filename. If you are permitted to create the file, minicom checks to see if it already exists. If so, it assumes it's a macro file and reads it in. If it isn't, well, it's your problem :-) If the file does not exist, the filename is accepted.

M - Edit Macros

opens up a new window which allows you to edit the F1 through F10 macros.

N - Macros enabled

- Yes or No. If macros are disabled, the F1-F10 keys will just send the VT100/VT220 function key escape sequences.

O - Character conversion

The active conversion table filename is shown here. If you can see no name, no conversion is active. Pressing O, you will see the conversion table edit menu.

Edit Macros

Here, the macros for F1 through F10 are defined. The bottom of the window shows a legend of character combinations that have special meaning. They allow you to enter special

control characters with plain text by prefixing them with a '^', in which '^^' means '^' itself. You can send a 1 second delay with the '^~' code. This is useful when you are trying to login after ftp'ing or telnet'ing somewhere. You can also include your current username and password from the phone directory in the macros with '\u' and '\p', respectively. If you need the backslash character in the macro, write it doubled as '\\'. To edit a macro, press the number (or letter for F10) and you will be moved to the end of the macro. When editing the line, you may use the left & right arrows, Home & End keys, Delete & BackSpace, and ESC and RETURN. ESC cancels any changes made while ENTER accepts the changes.

Character conversion

Here you can edit the character conversion table. If you are not an American, you know that in many languages there are characters that are not included in the ASCII character set, and in the old times they may have replaced some less important characters in ASCII and now they are often represented with character codes above 127. AND there are various different ways to represent them. This is where you may edit conversion tables for systems that use a character set different from the one on your computer.

A - Load table

You probably guessed it. This command loads a table from the disk. You are asked a file name for the table. Predefined tables .mciso, .mcpc8 and .mcsf7 should be included with the program. Table .mciso does no conversion, .mcpc8 is to be used for connections with systems that use the 8-bit pc character set, and .mcsf7 is for compatibility with the systems that uses the good old 7-bit coding to replace the characters {|}[]\ with the diacritical characters used in Finnish and Swedish.

B - Save table

This one saves the active table on the filename you specify.

C - edit char

This is where you can make your own modifications to the existing table. First you are asked the character value (in decimal) whose conversion you want to change. Next you'll say which character you want to see on your screen when that character comes from the outside world. And then you'll be asked what you want to be sent out when you enter that character from your keyboard.

D - next screen

E - prev screen

Yeah, you probably noticed that this screen shows you what kind of conversions are active. The screen just is (usually) too small to show the whole table at once in an easy-to-understand format. This is how you can scroll the table left and right.

F - convert capture

Toggles whether or not the character conversion table is used when writing the capture file.

Save setup as dfl

Save the parameters as the default for the next time the program is started. Instead of dfl, any other parameter name may appear, depending on which one was used when the program was started.

Save setup as..

Save the parameters under a special name. Whenever Minicom is started with this name as an argument, it will use these parameters. This option is of course priviliged to root.

Exit
Escape from this menu without saving. This can also be done with ESC.

Exit from minicom
Only root will see this menu entry, if he/she started minicom with the '-s' option. This way, it is possible to change the configuration without actually running minicom.

STATUS LINE

The status line has several indicators, that speak for themselves. The mysterious APP or NOR indicator probably needs explanation. The VT100 cursor keys can be in two modes: applications mode and cursor mode. This is controlled by an escape sequence. If you find that the cursor keys do not work in, say, vi when you're logged in using minicom then you can see with this indicator whether the cursor keys are in applications or cursor mode. You can toggle the two with the C-A I key. If the cursor keys then work, it's probably an error in the remote system's termcap initialization strings (is).

LOCALES

Minicom has now support for local languages. This means you can change most of the English messages and other strings to another language by setting the environment variable LANG. On September 2001 the supported languages are Brazilian Portuguese, Finnish, Japanese, French, Polish, Czech, Russian and Spanish. Turkish is under construction.

SECURITY ISSUES

Since Minicom is run setuid root on some computers, you probably want to restrict access to it. This is possible by using a configuration file in the same directory as the default files, called "minicom.users". The syntax of this file is as following:

<username> <configuration> [configuration...]

To allow user 'miquels' to use the default configuration, enter the following line into "minicom.users":

miquels dfl

If you want users to be able to use more than the default configurations, just add the names of those configurations behind the user name. If no configuration is given behind the username, minicom assumes that the user has access to all configurations.

MISC

If minicom is hung, kill it with SIGTERM . (This means kill -15, or since sigterm is default, just plain "kill <minicompid>". This will cause a graceful exit of minicom, doing resets and everything. You may kill minicom from a script with the command "! killall -9 minicom" without hanging up the line. Without the -9 parameter, minicom first hangs up before exiting.

Since a lot of escape sequences begin with ESC (Arrow up is ESC [A), Minicom does not know if the escape character it gets is you pressing the escape key, or part of a sequence.

An old version of Minicom, V1.2, solved this in a rather crude way: to get the escape key, you had to press it *twice*.

As of release 1.3 this has bettered a little: now a 1-second timeout is builtin, like in vi. For systems that have the select() system call the timeout is 0.5 seconds. And... surprise: a special Linux-dependant **hack** :-) was added. Now, minicom can separate the escape key and escape-sequences. To see how dirty this was done, look into wkeys.c. But it works like a charm!

FILES

Minicom keeps it's configuration files in one directory, usually /var/lib/minicom, /usr/local/etc or /etc. To find out what default directory minicom has compiled in, issue the command *minicom -h*. You'll probably also find the demo files for **runscript**(1), and the examples of character conversion tables either there or in the subdirectories of /usr/doc/minicom*. The conversion tables are named something like mc.* in that directory, but you probably want to copy the ones you need in your home directory as something beginning with a dot.

minicom.users
minirc.*
$HOME/.minirc.*
$HOME/.dialdir
$HOME/minicom.log
/usr/share/locale/*/LC_MESSAGES/minicom.mo

VERSION

Minicom is now up to version 2.1.

AUTHORS

The original author of minicom is Miquel van Smoorenburg (miquels@cistron.nl). He wrote versions up to 1.75.
Jukka Lahtinen (walker@netsonic.fi, jukkal@despammed.com) has been responsible for new versions since 1.78, helped by some other people, including:
filipg@paranoia.com wrote the History buffer searching to 1.79.
Arnaldo Carvalho de Melo (acme@conectiva.com.br) did the internationalization and the Brasilian Portuguese translations.
Jim Seymour (jseymour@jimsun.LinxNet.com) wrote the multiple modem support and the filename selection window used since 1.80.
Tomohiro Kubota (kubota@debian.or.jp) wrote the Japanese translations and the citation facility, and did some fixes.
Gael Queri (gqueri@mail.dotcom.fr) wrote the French translations.
Arkadiusz Miskiewicz (misiek@pld.org.pl) wrote the Polish translations.
Kim Soyoung (nexti@chollian.net) wrote the Korean translations.
Jork Loeser (jork.loeser@inf.tu-dresden.de) provided the socket extension.

Most of this man page is copied, with corrections, from the original minicom README, but some pieces and the corrections are by Michael K. Johnson.

Jukka Lahtinen (walker@netsonic.fi) has added some information of the changes made after version 1.75.

NAME

ascii-xfr − upload/download files using the ASCII protocol

SYNOPSIS

ascii-xfr -s|-r [**−ednv**] [**−l** *linedelay*] [**−c** *characterdelay*] *filename*

DESCRIPTION

Ascii-xfr Transfers files in ASCII mode. This means no flow control, no checksumming and no file-name negotiation. It should *only* be used if the remote system doesn't understand anything else.

The ASCII protocol transfers files line-by-line. The EOL (End-Of-Line) character is transmitted as CRLF. When receiving, the CR character is stripped from the incoming file. The Control-Z (ASCII 26) character signals End-Of-File, if option -e is specified (unless you change it to Cotrol-D (ASCII 4) with -d).

Ascii-xfr reads from *stdin* when receiving, and sends data on *stdout* when sending. Some form of input or output redirection to the the modem device is thus needed when downloading or uploading, respectively.

OPTIONS

−s Send a file.

−r Receive a file. One of **-s** or **-r** *must* be present.

−e Send the End-Of-File character (Control-Z, ASCII 26 by default) when uploading has finished.

−d Use the Control-D (ASCII 4) as End-Of-File character.

−n Do not translate CR to CRLF and vice versa.

−v Verbose: show tranfer statistics on the stderr output.

−l *milliseconds*
When transmitting, pause for this delay after each line.

−c *milliseconds*
When transmitting, pause for this delay after each character.

file Name of the file to send or receive. When receiving, any existing file by this name will be truncated.

USAGE WITH MINICOM

If you want to call this program from **minicom(1)**, start minicom and go to the **O**ptions menu. Select *File transfer protocols*. Add the following lines, for example as protocols **I** and **J**.

```
I Ascii   /usr/bin/ascii-xfr -sv  Y  U  N  Y
J Ascii   /usr/bin/ascii-xfr -rv  Y  D  N  Y
```

AUTHOR

Miquel van Smoorenburg, miquels@cistron.nl
Jukka Lahtinen, walker@netsonic.fi

SEE ALSO

minicom(1)

NAME

xminicom

SYNOPSIS

xminicom *minicom-options*

DESCRIPTION

Xminicom is a script wrapper around *minicom.* It tries to find a color capable xterm or rxvt on your system, and then runs minicom with the -c (color) flag in a terminal session.

OPTIONS

See *minicom* (1).

AUTHOR

Miquel van Smoorenburg, miquels@cistron.nl

SEE ALSO

minicom(1)

NAME

runscript − script interpreter for minicom

SYNOPSIS

runscript scriptname [logfile [homedir]]

DESCRIPTION

runscript is a simple script interpreter that can be called from within the minicom communications program to automate tasks like logging in to a unix system or your favorite bbs.

INVOCATION

The program expects a script name and optionally a filename and the user's home directory as arguments, and it expects that it's input and output are connected to the "remote end", the system you are connecting to. All messages from **runscript** ment for the local screen are directed to the **stderr** output. All this is automatically taken care of if you run it from **minicom**. The logfile and home directory parameters are only used to tell the log command the name of the logfile and where to write it. If the homedir is omitted, runscript uses the directory found in the $HOME environment variable. If also the logfile name is omitted, the log commands are ignored.

KEYWORDS

Runscript recognizes the following commands:

expect send goto gosub return !
exit print set inc dec if
timeout verbose sleep break call log

OVERVIEW OF KEYWORDS

send <string>

<string> is sent to the modem. It is followed by a '\r'. <string> can be:
- regular text, eg 'send hello'
- text enclosed in quotes, eg 'send "hello world"'

Within <string> the following sequences are recognized:
\n - newline
\r - carriage return
\a - bell
\b - backspace
\c - don't send the default '\r'.
\f - formfeed
\^ - the ^ character
\o - send character **o** (**o** is an octal number)

Control characters can be used in the string with the ^ prefix (^A to ^Z, ^[, ^ ^], ^^ and ^_). If you need to send the ^ chracter, you must prefix it with the \ escape character.
Also $(environment_variable) can be used, for example $(TERM). Minicom passes three special environment variables: $(LOGIN), which is the username, $(PASS), which is the password, as defined in the proper entry of the dialing directory, and $(TERMLIN) which is the number of actual terminal lines on your screen (that is, the statusline excluded).

print <string>

Prints <string> to the local screen. Default followed by '\r\n'. See the description of 'send' above.

label: Declares a label (with the name 'label') to use with goto or gosub.

goto <label>

Jump to another place in the program.

gosub <label>
Jumps to another place in the program. When the statement 'return' is encountered, control returns to the statement after the gosub. Gosub's can be nested.

return Return from a gosub.

! <command>
Runs a shell for you in which 'command' is executed. On return, the variable '$?' is set to the exit status of this command, so you can subsequently test it using 'if'.

exit [value]
Exit from "runscript" with an optional exit status. (default 1)

set <variable> <value>
Sets the value of <variable> (which is a single letter a-z) to the value <value>. If <variable> does not exist, it will be created. <value> can be a integer value or another variable.

inc <variable>
Increments the value of <variable> by one.

dec <variable>
Decrements the value of <variable> by one.

if <value> <operator> <value> <statement>
Conditional execution of <statement>. <operator> can be <, >, != or =. Eg, 'if a > 3 goto exitlabel'.

timeout <value>
Sets the global timeout. By default, 'runscript' will exit after 120 seconds. This can be changed with this command. Warning: this command acts differently within an 'expect' statement, but more about that later.

verbose <on|off>
By default, this is 'on'. That means that anything that is being read from the modem by 'runscript', gets echoed to the screen. This is so that you can see what 'runscript' is doing.

sleep <value>
Suspend execution for <value> seconds.

expect

```
expect {
  pattern  [statement]
  pattern  [statement]
  [timeout <value> [statement] ]
  ....
 }
```

The most important command of all. Expect keeps reading from the input until it reads a pattern that matches one of the specified ones. If expect encounters an optional statement after that pattern, it will execute it. Otherwise the default is to just break out of the expect. 'pattern' is a string, just as in 'send' (see above). Normally, expect will timeout in 60 seconds and just exit, but this can be changed with the timeout command.

break Break out of an 'expect' statement. This is normally only useful as argument to 'timeout' within an expect, because the default action of timeout is to exit immediately.

call <scriptname>
Transfers control to another scriptfile. When that scriptfile finishes without errors, the original script will continue.

log <text>
Write text to the logfile.

NOTES

If you want to make your script to exit minicom (for example when you use minicom to dial up your ISP, and then start a ppp or slip session from a script), try the command "! killall -9 minicom" as the last script command. The -9 option should prevent minicom from hanging up the line and resetting the modem before exiting.

Well, I don't think this is enough information to make you an experienced 'programmer' in 'runscript', but together with the examples it shouldn't be too hard to write some useful script files. Things will be easier if you have experience with BASIC. The **minicom** source code comes together with two example scripts, **scriptdemo** and **unixlogin**. Especially the last one is a good base to build on for your own scripts.

BUGS

Runscript should be built in to minicom.

AUTHOR

Miquel van Smoorenburg, <miquels@drinkel.ow.org> Jukka Lahtinen, <walker@netsonic.fi>

NAME

screen − screen manager with VT100/ANSI terminal emulation

SYNOPSIS

screen [−*options*] [*cmd* [*args*]]
screen −r [[*pid***.**]*tty*[**.***host*]]
screen −r *sessionowner/*[[*pid***.**]*tty*[**.***host*]]

DESCRIPTION

Screen is a full-screen window manager that multiplexes a physical terminal between several processes (typically interactive shells). Each virtual terminal provides the functions of a DEC VT100 terminal and, in addition, several control functions from the ISO 6429 (ECMA 48, ANSI X3.64) and ISO 2022 standards (e. g. insert/delete line and support for multiple character sets). There is a scrollback history buffer for each virtual terminal and a copy-and-paste mechanism that allows moving text regions between windows.

When *screen* is called, it creates a single window with a shell in it (or the specified command) and then gets out of your way so that you can use the program as you normally would. Then, at any time, you can create new (full-screen) windows with other programs in them (including more shells), kill existing windows, view a list of windows, turn output logging on and off, copy-and-paste text between windows, view the scrollback history, switch between windows in whatever manner you wish, etc. All windows run their programs completely independent of each other. Programs continue to run when their window is currently not visible and even when the whole *screen* session is detached from the user's terminal. When a program terminates, *screen* (per default) kills the window that contained it. If this window was in the foreground, the display switches to the previous window; if none are left, *screen* exits.

Everything you type is sent to the program running in the current window. The only exception to this is the one keystroke that is used to initiate a command to the window manager. By default, each command begins with a control-a (abbreviated C-a from now on), and is followed by one other keystroke. The command character and all the key bindings can be fully customized to be anything you like, though they are always two characters in length.

Screen does not understand the prefix ''C-'' to mean control. Please use the caret notation (''^A'' instead of ''C-a'') as arguments to e.g. the *escape* command or the *-e* option. *Screen* will also print out control characters in caret notation.

The standard way to create a new window is to type ''C-a c''. This creates a new window running a shell and switches to that window immediately, regardless of the state of the process running in the current window. Similarly, you can create a new window with a custom command in it by first binding the command to a keystroke (in your .screenrc file or at the ''C-a :'' command line) and then using it just like the ''C-a c'' command. In addition, new windows can be created by running a command like:

```
screen emacs prog.c
```

from a shell prompt within a previously created window. This will not run another copy of *screen*, but will instead supply the command name and its arguments to the window manager (specified in the $STY environment variable) who will use it to create the new window. The above example would start the emacs editor (editing prog.c) and switch to its window.

If ''/etc/utmp'' is writable by *screen*, an appropriate record will be written to this file for each window, and removed when the window is terminated. This is useful for working with ''talk'', ''script'', ''shutdown'', ''rsend'', ''sccs'' and other similar programs that use the utmp file to determine who you are. As long as *screen* is active on your terminal, the terminal's own record is removed from the utmp file. See also ''C-a L''.

GETTING STARTED

Before you begin to use *screen* you'll need to make sure you have correctly selected your terminal type, just as you would for any other termcap/terminfo program. (You can do this by using *tset* for example.)

If you're impatient and want to get started without doing a lot more reading, you should remember this one command: ''C-a ?''. Typing these two characters will display a list of the available *screen* commands and their bindings. Each keystroke is discussed in the section ''DEFAULT KEY BINDINGS''. The manual section ''CUSTOMIZATION'' deals with the contents of your .screenrc.

If your terminal is a ''true'' auto-margin terminal (it doesn't allow the last position on the screen to be updated without scrolling the screen) consider using a version of your terminal's termcap that has automatic margins turned *off*. This will ensure an accurate and optimal update of the screen in all circumstances. Most terminals nowadays have ''magic'' margins (automatic margins plus usable last column). This is the VT100 style type and perfectly suited for *screen*. If all you've got is a ''true'' auto-margin terminal *screen* will be content to use it, but updating a character put into the last position on the screen may not be possible until the screen scrolls or the character is moved into a safe position in some other way. This delay can be shortened by using a terminal with insert-character capability.

COMMAND-LINE OPTIONS

Screen has the following command-line options:

−a include *all* capabilities (with some minor exceptions) in each window's termcap, even if *screen* must redraw parts of the display in order to implement a function.

−A Adapt the sizes of all windows to the size of the current terminal. By default, *screen* tries to restore its old window sizes when attaching to resizable terminals (those with ''WS'' in its description, e.g. suncmd or some xterm).

−c *file*
override the default configuration file from ''$HOME/.screenrc'' to *file*.

−d|**−D** [*pid.tty.host*]
does not start *screen*, but detaches the elsewhere running *screen* session. It has the same effect as typing ''C-a d'' from *screen*'s controlling terminal. **−D** is the equivalent to the power detach key. If no session can be detached, this option is ignored. In combination with the **−r**/**−R** option more powerful effects can be achieved:

−d −r Reattach a session and if necessary detach it first.

−d −R Reattach a session and if necessary detach or even create it first.

−d −RR Reattach a session and if necessary detach or create it. Use the first session if more than one session is available.

−D −r Reattach a session. If necessary detach and logout remotely first.

−D −R Attach here and now. In detail this means: If a session is running, then reattach. If necessary detach and logout remotely first. If it was not running create it and notify the user. This is the author's favorite.

−D −RR
Attach here and now. Whatever that means, just do it.

Note: It is always a good idea to check the status of your sessions by means of ''screen −list''.

−e *xy*
specifies the command character to be *x* and the character generating a literal command character to *y* (when typed after the command character). The default is ''C-a'' and 'a', which can be specified as ''-eˆAa''. When creating a *screen* session, this option sets the default command character. In a multiuser session all users added will start off with this command character. But when attaching to an already running session, this option changes only the command character of the attaching user. This

option is equivalent to either the commands ‘‘defescape’’ or ‘‘escape’’ respectively.

−f, **−fn**, and **−fa**
turns flow-control on, off, or ‘‘automatic switching mode’’. This can also be defined through the ‘‘defflow’’ .screenrc command.

−h *num*
Specifies the history scrollback buffer to be *num* lines high.

−i will cause the interrupt key (usually C-c) to interrupt the display immediately when flow-control is on. See the ‘‘defflow’’ .screenrc command for details. The use of this option is discouraged.

−l and **−ln**
turns login mode on or off (for /etc/utmp updating). This can also be defined through the ‘‘deflogin’’ .screenrc command.

−ls and **−list**
does not start *screen*, but prints a list of *pid.tty.host* strings identifying your *screen* sessions. Sessions marked ‘detached’ can be resumed with ‘‘screen -r’’. Those marked ‘attached’ are running and have a controlling terminal. If the session runs in multiuser mode, it is marked ‘multi’. Sessions marked as ‘unreachable’ either live on a different host or are ‘dead’. An unreachable session is considered dead, when its name matches either the name of the local host, or the specified parameter, if any. See the **-r** flag for a description how to construct matches. Sessions marked as ‘dead’ should be thoroughly checked and removed. Ask your system administrator if you are not sure. Remove sessions with the **-wipe** option.

−L tells *screen* to turn on automatic output logging for the windows.

−m causes *screen* to ignore the $STY environment variable. With ‘‘screen -m’’ creation of a new session is enforced, regardless whether *screen* is called from within another *screen* session or not. This flag has a special meaning in connection with the ‘-d’ option:

−d −m Start *screen* in ‘‘detached’’ mode. This creates a new session but doesn’t attach to it. This is useful for system startup scripts.

−D −m This also starts screen in ‘‘detached’’ mode, but doesn’t fork a new process. The command exits if the session terminates.

−O selects a more optimal output mode for your terminal rather than true VT100 emulation (only affects auto-margin terminals without ‘LP’). This can also be set in your .screenrc by specifying ‘OP’ in a ‘‘termcap’’ command.

−p *number_or_name*
Preselect a window. This is usefull when you want to reattach to a specific windor or you want to send a command via the ‘‘-X’’ option to a specific window. As with screen’s select commant, ‘‘-’’ selects the blank window. As a special case for reattach, ‘‘=’’ brings up the windowlist on the blank window.

−q Suppress printing of error messages. In combination with ‘‘-ls’’ the exit value is as follows: 9 indicates a directory without sessions. 10 indicates a directory with running but not attachable sessions. 11 (or more) indicates 1 (or more) usable sessions. In combination with ‘‘-r’’ the exit value is as follows: 10 indicates that there is no session to resume. 12 (or more) indicates that there are 2 (or more) sessions to resume and you should specify which one to choose. In all other cases ‘‘-q’’ has no effect.

−r [*pid.tty.host*]
−r *sessionowner/[pid.tty.host*]
resumes a detached *screen* session. No other options (except combinations with **−d**/**−D**) may be specified, though an optional prefix of [*pid.*]*tty.host* may be needed to distinguish between multiple detached *screen* sessions. The second form is used to connect to another user’s screen session which runs in multiuser mode. This indicates that screen should look for sessions in another user’s directory. This requires setuid-root.

−R attempts to resume the first detached *screen* session it finds. If successful, all other command-line options are ignored. If no detached session exists, starts a new session using the specified options, just as if **−R** had not been specified. The option is set by default if *screen* is run as a login-shell (actually screen uses ‘‘-xRR’’ in that case). For combinations with the **−d**/**−D** option see there.

−s sets the default shell to the program specified, instead of the value in the environment variable $SHELL (or ‘‘/bin/sh’’ if not defined). This can also be defined through the ‘‘shell’’ .screenrc command.

−S *sessionname*
When creating a new session, this option can be used to specify a meaningful name for the session. This name identifies the session for ‘‘screen -list’’ and ‘‘screen -r’’ actions. It substitutes the default [*tty.host*] suffix.

−t *name*
sets the title (a. k. a.) for the default shell or specified program. See also the ‘‘shelltitle’’ .screenrc command.

−U Run screen in UTF-8 mode. This option tells screen that your terminal sends and understands UTF-8 encoded characters. It also sets the default encoding for new windows to ‘utf8’.

−v Print version number.

−wipe [*match*]
does the same as ‘‘screen -ls’’, but removes destroyed sessions instead of marking them as ‘dead’. An unreachable session is considered dead, when its name matches either the name of the local host, or the explicitly given parameter, if any. See the **-r** flag for a description how to construct matches.

−x Attach to a not detached *screen* session. (Multi display mode).

−X Send the specified command to a running screen session. You can use the **-d** or **-r** option to tell screen to look only for attached or detached screen sessions. Note that this command doesn’t work if the session is password protected.

DEFAULT KEY BINDINGS

As mentioned, each *screen* command consists of a ‘‘C-a’’ followed by one other character. For your convenience, all commands that are bound to lower-case letters are also bound to their control character counterparts (with the exception of ‘‘C-a a’’; see below), thus, ‘‘C-a c’’ as well as ‘‘C-a C-c’’ can be used to create a window. See section ‘‘CUSTOMIZATION’’ for a description of the command.

The following table shows the default key bindings:

C-a ’	(select)	Prompt for a window name or number to switch to.
C-a "	(windowlist -b)	Present a list of all windows for selection.
C-a 0	(select 0)	
...	...	
C-a 9	(select 9)	
C-a -	(select -)	Switch to window number 0 − 9, or to the blank window.
C-a tab	(focus)	Switch the input focus to the next region.
C-a C-a	(other)	Toggle to the window displayed previously. Note that this binding defaults to the command character typed twice, unless overridden. For instance, if you use the option ‘‘**−e]x**’’, this command becomes ‘‘]]’’.
C-a a	(meta)	Send the command character (C-a) to window. See *escape* command.
C-a A	(title)	Allow the user to enter a name for the current window.
C-a b		

C-a C-b	(break)	Send a break to window.
C-a B	(pow_break)	Reopen the terminal line and send a break.
C-a c **C-a C-c**	(screen)	Create a new window with a shell and switch to that window.
C-a C	(clear)	Clear the screen.
C-a d **C-a C-d**	(detach)	Detach *screen* from this terminal.
C-a D D	(pow_detach)	Detach and logout.
C-a f **C-a C-f**	(flow)	Toggle flow *on*, *off* or *auto*.
C-a F	(fit)	Resize the window to the current region size.
C-a C-g	(vbell)	Toggles *screen's* visual bell mode.
C-a h	(hardcopy)	Write a hardcopy of the current window to the file ''hardcopy.*n*''.
C-a H	(log)	Begins/ends logging of the current window to the file ''screenlog.*n*''.
C-a i **C-a C-i**	(info)	Show info about this window.
C-a k **C-a C-k**	(kill)	Destroy current window.
C-a l **C-a C-l**	(redisplay)	Fully refresh current window.
C-a L	(login)	Toggle this windows login slot. Available only if *screen* is configured to update the utmp database.
C-a m **C-a C-m**	(lastmsg)	Repeat the last message displayed in the message line.
C-a M	(monitor)	Toggles monitoring of the current window.
C-a space **C-a n** **C-a C-n**	(next)	Switch to the next window.
C-a N	(number)	Show the number (and title) of the current window.
C-a backspace **C-a h** **C-a p** **C-a C-p**	(prev)	Switch to the previous window (opposite of **C-a n**).
C-a q **C-a C-q**	(xon)	Send a control-q to the current window.
C-a Q	(only)	Delete all regions but the current one.
C-a r **C-a C-r**	(wrap)	Toggle the current window's line-wrap setting (turn the current window's automatic margins on and off).
C-a s **C-a C-s**	(xoff)	Send a control-s to the current window.
C-a S	(split)	Split the current region into two new ones.
C-a t		

C-a C-t	(time)	Show system information.
C-a v	(version)	Display the version and compilation date.
C-a C-v	(digraph)	Enter digraph.
C-a w **C-a C-w**	(windows)	Show a list of window.
C-a W	(width)	Toggle 80/132 columns.
C-a x **C-a C-x**	(lockscreen)	Lock this terminal.
C-a X	(remove)	Kill the current region.
C-a z **C-a C-z**	(suspend)	Suspend *screen*. Your system must support BSD-style job-control.
C-a Z	(reset)	Reset the virtual terminal to its ‘‘power-on’’ values.
C-a .	(dumptermcap)	Write out a ‘‘.termcap’’ file.
C-a ?	(help)	Show key bindings.
C-a C-	(quit)	Kill all windows and terminate *screen*.
C-a :	(colon)	Enter command line mode.
C-a [**C-a C-[** **C-a esc**	(copy)	Enter copy/scrollback mode.
C-a]	(paste .)	Write the contents of the paste buffer to the stdin queue of the current window.
C-a { **C-a }**	(history)	Copy and paste a previous (command) line.
C-a >	(writebuf)	Write paste buffer to a file.
C-a <	(readbuf)	Reads the screen-exchange file into the paste buffer.
C-a =	(removebuf)	Removes the file used by **C-a <** and **C-a >**.
C-a ,	(license)	Shows where *screen* comes from, where it went to and why you can use it.
C-a _	(silence)	Start/stop monitoring the current window for inactivity.
C-a *	(displays)	Show a listing of all currently attached displays.

CUSTOMIZATION

The ‘‘socket directory’’ defaults either to $HOME/.screen or simply to /tmp/screens or preferably to /usr/local/screens chosen at compile-time. If *screen* is installed setuid-root, then the administrator should compile *screen* with an adequate (not NFS mounted) socket directory. If *screen* is not running setuid-root, the user can specify any mode 700 directory in the environment variable $SCREENDIR.

When *screen* is invoked, it executes initialization commands from the files ‘‘/usr/local/etc/screenrc’’ and ‘‘.screenrc’’ in the user’s home directory. These are the ‘‘programmer’s defaults’’ that can be overridden in the following ways: for the global screenrc file *screen* searches for the environment variable $SYSSCREENRC (this override feature may be disabled at compile-time). The user specific screenrc file is searched in $SCREENRC, then $HOME/.screenrc. The command line option **-c** takes precedence over the above user screenrc files.

Commands in these files are used to set options, bind functions to keys, and to automatically establish one or more windows at the beginning of your *screen* session. Commands are listed one per line, with empty lines being ignored. A command’s arguments are separated by tabs or spaces, and may be surrounded by

single or double quotes. A ‘#’ turns the rest of the line into a comment, except in quotes. Unintelligible lines are warned about and ignored. Commands may contain references to environment variables. The syntax is the shell-like "$VAR " or "${VAR}". Note that this causes incompatibility with previous *screen* versions, as now the ’$’-character has to be protected with ’\’ if no variable substitution shall be performed. A string in single-quotes is also protected from variable substitution.

Two configuration files are shipped as examples with your screen distribution: ‘‘etc/screenrc’’ and ‘‘etc/etc-screenrc’’. They contain a number of useful examples for various commands.

Customization can also be done ’on-line’. To enter the command mode type ‘C-a :’. Note that commands starting with ‘‘def’’ change default values, while others change current settings.

The following commands are available:

acladd *usernames* [*crypted-pw*]
addacl *usernames*

Enable users to fully access this screen session. *Usernames* can be one user or a comma separated list of users. This command enables to attach to the *screen* session and performs the equivalent of ‘aclchg *usernames* +rwx "#?"’. executed. To add a user with restricted access, use the ‘aclchg’ command below. If an optional second parameter is supplied, it should be a crypted password for the named user(s). ‘Addacl’ is a synonym to ‘acladd’. Multi user mode only.

aclchg *usernames permbits list*
chacl *usernames permbits list*

Change permissions for a comma separated list of users. Permission bits are represented as ‘r’, ‘w’ and ‘x’. Prefixing ‘+’ grants the permission, ‘-’ removes it. The third parameter is a comma separated list of commands and/or windows (specified either by number or title). The special list ‘#’ refers to all windows, ‘?’ to all commands. if *usernames* consists of a single ‘*’, all known users are affected. A command can be executed when the user has the ‘x’ bit for it. The user can type input to a window when he has its ‘w’ bit set and no other user obtains a writelock for this window. Other bits are currently ignored. To withdraw the writelock from another user in window 2: ‘aclchg *username* -w+w 2’. To allow read-only access to the session: ‘aclchg *username* -w "#"’. As soon as a user’s name is known to *screen* he can attach to the session and (per default) has full permissions for all command and windows. Execution permission for the acl commands, ‘at’ and others should also be removed or the user may be able to regain write permission. Rights of the special username **nobody** cannot be changed (see the ‘‘su’’ command). ‘Chacl’ is a synonym to ‘aclchg’. Multi user mode only.

acldel *username*

Remove a user from *screen*’s access control list. If currently attached, all the user’s displays are detached from the session. He cannot attach again. Multi user mode only.

aclgrp *username* [*groupname*]

Creates groups of users that share common access rights. The name of the group is the username of the group leader. Each member of the group inherits the permissions that are granted to the group leader. That means, if a user fails an access check, another check is made for the group leader. A user is removed from all groups the special value ‘‘none’’ is used for *groupname*. If the second parameter is omitted all groups the user is in are listed.

aclumask [[*users*]+*bits* |[*users*]-*bits* ....]
umask [[*users*]+*bits* |[*users*]-*bits* ....]

This specifies the access other users have to windows that will be created by the caller of the command. *Users* may be no, one or a comma separated list of known usernames. If no users are specified, a list of all currently known users is assumed. *Bits* is any combination of access control bits allowed defined with the ‘‘aclchg’’ command. The special username ‘‘?’’ predefines the access that not yet known users will be

granted to any window initially. The special username ‘‘??’’ predefines the access that not yet known users are granted to any command. Rights of the special username **nobody** cannot be changed (see the ‘‘su’’ command). ‘Umask’ is a synonym to ‘aclumask’.

activity *message*

When any activity occurs in a background window that is being monitored, *screen* displays a notification in the message line. The notification message can be re-defined by means of the ‘‘activity’’ command. Each occurrence of ‘%’ in *message* is replaced by the number of the window in which activity has occurred, and each occurrence of ‘^G’ is replaced by the definition for bell in your termcap (usually an audible bell). The default message is

’Activity in window %n’

Note that monitoring is off for all windows by default, but can be altered by use of the ‘‘monitor’’ command (C-a M).

allpartial on|off

If set to on, only the current cursor line is refreshed on window change. This affects all windows and is useful for slow terminal lines. The previous setting of full/partial refresh for each window is restored with ‘‘allpartial off’’. This is a global flag that immediately takes effect on all windows overriding the ‘‘partial’’ settings. It does not change the default redraw behavior of newly created windows.

altscreen on|off

If set to on, "alternate screen" support is enabled in virtual terminals, just like in xterm. Initial setting is ‘off’.

at [*identifier*][**#**|*****|**%**] *command* [*args* ...]

Execute a command at other displays or windows as if it had been entered there. ‘‘At’’ changes the context (the ‘current window’ or ‘current display’ setting) of the command. If the first parameter describes a non-unique context, the command will be executed multiple times. If the first parameter is of the form ‘*identifier**’ then identifier is matched against user names. The command is executed once for each display of the selected user(s). If the first parameter is of the form ‘*identifier*%’ identifier is matched against displays. Displays are named after the ttys they attach. The prefix ‘/dev/’ or ‘/dev/tty’ may be omitted from the identifier. If *identifier* has a ‘#’ or nothing appended it is matched against window numbers and titles. Omitting an identifier in front of the ‘#’, ‘*’ or ‘%’-character selects all users, displays or windows because a prefix-match is performed. Note that on the affected display(s) a short message will describe what happened. Permission is checked for initiator of the ‘‘at’’ command, not for the owners of the affected display(s). Note that the ’#’ character works as a comment introducer when it is preceded by whitespace. This can be escaped by prefixing a ’\’. Permission is checked for the initiator of the ‘‘at’’ command, not for the owners of the affected display(s).
Caveat: When matching against windows, the command is executed at least once per window. Commands that change the internal arrangement of windows (like ‘‘other’’) may be called again. In shared windows the command will be repeated for each attached display. Beware, when issuing toggle commands like ‘‘login’’! Some commands (e.g. ‘‘process’’) require that a display is associated with the target windows. These commands may not work correctly under ‘‘at’’ looping over windows.

attrcolor *attrib* [*attribute/color-modifier*]

This command can be used to highlight attributes by changing the color of the text. If the attribute *attrib* is in use, the specified attribute/color modifier is also applied. If no modifier is given, the current one is deleted. See the ‘‘STRING ESCAPES’’ chapter for the syntax of the modifier. Screen understands two pseudo-attributes, ‘‘i’’ stands for high-intensity foreground color and ‘‘I’’ for high-intensity background color.

Examples:

attrcolor b "R"

Change the color to bright red if bold text is to be printed.

attrcolor u "-u b"

Use blue text instead of underline.

attrcolor b ".I"

Use bright colors for bold text. Most terminal emulators do this already.

attrcolor i "+b"

Make bright colored text also bold.

autodetach on|off

Sets whether *screen* will automatically detach upon hangup, which saves all your running programs until they are resumed with a **screen -r** command. When turned off, a hangup signal will terminate *screen* and all the processes it contains. Autodetach is on by default.

autonuke on|off

Sets whether a clear screen sequence should nuke all the output that has not been written to the terminal. See also ‘‘obuflimit’’.

backtick *id lifespan autorefresh cmd args...*
backtick *id*

Program the backtick command with the numerical id *id*. The output of such a command is used for substitution of the ‘‘%‘’ string escape. The specified *lifespan* is the number of seconds the output is considered valid. After this time, the command is run again if a corresponding string escape is encountered. The *autorefresh* parameter triggers an automatic refresh for caption and hardstatus strings after the specified number of seconds. Only the last line of output is used for substitution.
If both the *lifespan* and the *autorefresh* parameters are zero, the backtick program is expected to stay in the background and generate output once in a while. In this case, the command is executed right away and screen stores the last line of output. If a new line gets printed screen will automatically refresh the hardstatus or the captions.
The second form of the command deletes the backtick command with the numerical id *id*.

bce [**on**|**off**]

Change background-color-erase setting. If ‘‘bce’’ is set to on, all characters cleared by an erase/insert/scroll/clear operation will be displayed in the current background color. Otherwise the default background color is used.

bell_msg [*message*]

When a bell character is sent to a background window, *screen* displays a notification in the message line. The notification message can be re-defined by this command. Each occurrence of ‘%’ in *message* is replaced by the number of the window to which a bell has been sent, and each occurrence of ‘^G’ is replaced by the definition for bell in your termcap (usually an audible bell). The default message is

’Bell in window %n’

An empty message can be supplied to the ‘‘bell_msg’’ command to suppress output of a message line (bell_msg ""). Without parameter, the current message is shown.

bind [**-c** *class*] *key* [*command* [*args*]]

Bind a command to a key. By default, most of the commands provided by *screen* are bound to one or more keys as indicated in the ‘‘DEFAULT KEY BINDINGS’’ section, e. g. the command to create a new window is bound to ‘‘C-c’’ and ‘‘c’’. The ‘‘bind’’ command can be used to redefine the key bindings and to define new bindings. The *key* argument is either a single character, a two-character sequence of the form ‘‘^x’’ (meaning ‘‘C-x’’), a backslash followed by an octal number (specifying the ASCII code of the character), or a backslash followed by a second character, such as ‘‘\^’’ or ‘‘\\’’. The argument can also be quoted, if you like. If no further argument is given, any previously established binding for this key is removed. The *command* argument can be any command listed in this section.

If a command class is specified via the ‘‘-c’’ option, the key is bound for the specified class. Use the ‘‘command’’ command to activate a class. Command classes can be used to create multiple command keys or multi-character bindings.

Some examples:

```
bind ' ' windows
bind ^k
bind k
bind K kill
bind ^f screen telnet foobar
bind \033 screen -ln -t root -h 1000 9 su
```

would bind the space key to the command that displays a list of windows (so that the command usually invoked by ‘‘C-a C-w’’ would also be available as ‘‘C-a space’’). The next three lines remove the default kill binding from ‘‘C-a C-k’’ and ‘‘C-a k’’. ‘‘C-a K’’ is then bound to the kill command. Then it binds ‘‘C-f’’ to the command ‘‘create a window with a TELNET connection to foobar’’, and bind ‘‘escape’’ to the command that creates an non-login window with a. k. a. ‘‘root’’ in slot #9, with a superuser shell and a scrollback buffer of 1000 lines.

```
bind -c demo1 0 select 10
bind -c demo1 1 select 11
bind -c demo1 2 select 12
bindkey "^B" command -c demo1
```

makes ‘‘C-b 0’’ select window 10, ‘‘C-b 1’’ window 11, etc.

```
bind -c demo2 0 select 10
bind -c demo2 1 select 11
bind -c demo2 2 select 12
bind - command -c demo2
```

makes ‘‘C-a - 0’’ select window 10, ‘‘C-a - 1’’ window 11, etc.

bindkey [**-d**] [**-m**] [**-a**] [[**-k**|**-t**] *string* [*cmd args*]]

This command manages screen’s input translation tables. Every entry in one of the tables tells screen how to react if a certain sequence of characters is encountered. There are three tables: one that should contain actions programmed by the user, one for the default actions used for terminal emulation and one for screen’s copy mode to do cursor movement. See section ‘‘INPUT TRANSLATION’’ for a list of default key bindings.
If the **-d** option is given, bindkey modifies the default table, **-m** changes the copy mode table and with neither option the user table is selected. The argument *string* is the sequence of characters to which an action is bound. This can either be a fixed string or a termcap keyboard capability name (selectable with the **-k** option).
Some keys on a VT100 terminal can send a different string if application mode is turned on (e.g the cursor keys). Such keys have two entries in the translation table. You can select the application mode entry by specifying the **-a** option.
The **-t** option tells screen not to do inter-character timing. One cannot turn off the timing if a termcap

capability is used.
Cmd can be any of screen's commands with an arbitrary number of *args*. If *cmd* is omitted the key-binding is removed from the table.
Here are some examples of keyboard bindings:

bindkey -d
Show all of the default key bindings. The application mode entries are marked with [A].

bindkey -k k1 select 1
Make the "F1" key switch to window one.

bindkey -t foo stuff barfoo
Make "foo" an abbreviation of the word "barfoo". Timeout is disabled so that users can type slowly.

bindkey "\024" mapdefault
This key-binding makes ''^T'' an escape character for key-bindings. If you did the above ''stuff barfoo'' binding, you can enter the word ''foo'' by typing ''^Tfoo''. If you want to insert a ''^T'' you have to press the key twice (i.e. escape the escape binding).

bindkey -k F1 command
Make the F11 (not F1!) key an alternative screen escape (besides ^A).

break [*duration*]

Send a break signal for *duration**0.25 seconds to this window. For non-Posix systems the time interval may be rounded up to full seconds. Most useful if a character device is attached to the window rather than a shell process (See also chapter ''WINDOW TYPES''). The maximum duration of a break signal is limited to 15 seconds.

blanker

Activate the screen blanker. First the screen is cleared. If no blanker program is defined, the cursor is turned off, otherwise, the program is started and it's output is written to the screen. The screen blanker is killed with the first keypress, the read key is discarded.
This command is normally used together with the ''idle'' command.

blankerprg [*program args*]

Defines a blanker program. Disables the blanker program if no arguments are given.

breaktype [*tcsendbreak*|*TIOCSBRK* |*TCSBRK*]

Choose one of the available methods of generating a break signal for terminal devices. This command should affect the current window only. But it still behaves identical to ''defbreaktype''. This will be changed in the future. Calling ''breaktype'' with no parameter displays the break method for the current window.

bufferfile [*exchange-file*]

Change the filename used for reading and writing with the paste buffer. If the optional argument to the ''bufferfile'' command is omitted, the default setting (''/tmp/screen-exchange'') is reactivated. The following example will paste the system's password file into the *screen* window (using the paste buffer, where a copy remains):

C-a : bufferfile /etc/passwd
C-a < C-a]
C-a : bufferfile

c1 [**on**|**off**]

Change c1 code processing. ‘‘C1 on’’ tells screen to treat the input characters between 128 and 159 as control functions. Such an 8-bit code is normally the same as ESC followed by the corresponding 7-bit code. The default setting is to process c1 codes and can be changed with the ‘‘defc1’’ command. Users with fonts that have usable characters in the c1 positions may want to turn this off.

caption always|**splitonly** [*string*]
caption string [*string*]

This command controls the display of the window captions. Normally a caption is only used if more than one window is shown on the display (split screen mode). But if the type is set to **always** screen shows a caption even if only one window is displayed. The default is **splitonly**.

The second form changes the text used for the caption. You can use all escapes from the ‘‘STRING ESCAPES’’ chapter. Screen uses a default of ‘%3n %t’.

You can mix both forms by providing a string as an additional argument.

charset *set*

Change the current character set slot designation and charset mapping. The first four character of *set* are treated as charset designators while the fifth and sixth character must be in range ’0’ to ’3’ and set the GL/GR charset mapping. On every position a ’.’ may be used to indicate that the corresponding charset/mapping should not be changed (*set* is padded to six characters internally by appending ’.’ chars). New windows have "BBBB02" as default charset, unless a ‘‘encoding’’ command is active.
The current setting can be viewed with the ‘‘info’’ command.

chdir [*directory*]

Change the *current directory* of *screen* to the specified directory or, if called without an argument, to your home directory (the value of the environment variable $HOME). All windows that are created by means of the ‘‘screen’’ command from within ‘‘.screenrc’’ or by means of ‘‘C-a : screen ...’’ or ‘‘C-a c’’ use this as their default directory. Without a chdir command, this would be the directory from which *screen* was invoked. Hardcopy and log files are always written to the *window’s* default directory, *not* the current directory of the process running in the window. You can use this command multiple times in your .screenrc to start various windows in different default directories, but the last chdir value will affect all the windows you create interactively.

clear

Clears the current window and saves its image to the scrollback buffer.

colon [*prefix*]

Allows you to enter ‘‘.screenrc’’ command lines. Useful for on-the-fly modification of key bindings, specific window creation and changing settings. Note that the ‘‘set’’ keyword no longer exists! Usually commands affect the current window rather than default settings for future windows. Change defaults with commands starting with ’def...’.

If you consider this as the ‘Ex command mode’ of *screen*, you may regard ‘‘C-a esc’’ (copy mode) as its ‘Vi command mode’.

command [**-c** *class*]

This command has the same effect as typing the screen escape character (^A). It is probably only useful for key bindings. If the ‘‘-c’’ option is given, select the specified command class. See also ‘‘bind’’ and ‘‘bindkey’’.

compacthist [**on**|**off**]

This tells screen whether to suppress trailing blank lines when scrolling up text into the history buffer.

console [**on**|**off**]

Grabs or un-grabs the machines console output to a window. *Note*: Only the owner of /dev/console can grab the console output. This command is only available if the machine supports the ioctl TIOCCONS.

copy

Enter copy/scrollback mode. This allows you to copy text from the current window and its history into the paste buffer. In this mode a vi-like 'full screen editor' is active:
Movement keys:
h, **j**, **k**, **l** move the cursor line by line or column by column.
0, **^** and **$** move to the leftmost column, to the first or last non-whitespace character on the line.
H, **M** and **L** move the cursor to the leftmost column of the top, center or bottom line of the window.
+ and **−** positions one line up and down.
G moves to the specified absolute line (default: end of buffer).
| moves to the specified absolute column.
w, **b**, **e** move the cursor word by word.
B, **E** move the cursor WORD by WORD (as in vi).
C-u and **C-d** scroll the display up/down by the specified amount of lines while preserving the cursor position. (Default: half screen-full).
C-b and **C-f** scroll the display up/down a full screen.
g moves to the beginning of the buffer.
% jumps to the specified percentage of the buffer.

Note:
Emacs style movement keys can be customized by a .screenrc command. (E. g. markkeys "h=^B:l=^F:$=^E") There is no simple method for a full emacs-style keymap, as this involves multi-character codes.

Marking:
The copy range is specified by setting two marks. The text between these marks will be highlighted. Press
space to set the first or second mark respectively.
Y and **y** used to mark one whole line or to mark from start of line.
W marks exactly one word.
Repeat count:
Any of these commands can be prefixed with a repeat count number by pressing digits
0..**9** which is taken as a repeat count.
Example: ''C-a C-[H 10 j 5 Y'' will copy lines 11 to 15 into the paste buffer.
Searching:
/ *Vi*-like search forward.
? *Vi*-like search backward.
C-a s *Emacs* style incremental search forward.
C-r *Emacs* style reverse i-search.
Specials:
There are however some keys that act differently than in *vi*. *Vi* does not allow one to yank rectangular blocks of text, but *screen* does. Press
c or **C** to set the left or right margin respectively. If no repeat count is given, both default to the current cursor position.
Example: Try this on a rather full text screen: ''C-a [M 20 l SPACE c 10 l 5 j C SPACE''.

This moves one to the middle line of the screen, moves in 20 columns left, marks the beginning of the

paste buffer, sets the left column, moves 5 columns down, sets the right column, and then marks the end of the paste buffer. Now try:
''C-a [M 20 l SPACE 10 l 5 j SPACE''

and notice the difference in the amount of text copied.

J joins lines. It toggles between 4 modes: lines separated by a newline character (012), lines glued seamless, lines separated by a single whitespace and comma separated lines. Note that you can prepend the newline character with a carriage return character, by issuing a ''crlf on''.

v is for all the *vi* users with '':set numbers'' − it toggles the left margin between column 9 and 1. Press

a before the final space key to toggle in append mode. Thus the contents of the paste buffer will not be overwritten, but is appended to.

A toggles in append mode and sets a (second) mark.

> sets the (second) mark and writes the contents of the paste buffer to the screen-exchange file (/tmp/screen-exchange per default) once copy-mode is finished.
This example demonstrates how to dump the whole scrollback buffer to that file: ''C-A [g SPACE G $ >''.

C-g gives information about the current line and column.

x exchanges the first mark and the current cursor position. You can use this to adjust an already placed mark.

@ does nothing. Does not even exit copy mode.

All keys not described here exit copy mode.

copy_reg [*key*]

No longer exists, use ''readreg'' instead.

crlf [**on**|**off**]

This affects the copying of text regions with the 'C-a [' command. If it is set to 'on', lines will be separated by the two character sequence 'CR' - 'LF'. Otherwise (default) only 'LF' is used. When no parameter is given, the state is toggled.

debug on|**off**

Turns runtime debugging on or off. If *screen* has been compiled with option -DDEBUG debugging available and is turned on per default. Note that this command only affects debugging output from the main ''SCREEN'' process correctly. Debug output from attacher processes can only be turned off once and forever.

defc1 on|**off**

Same as the **c1** command except that the default setting for new windows is changed. Initial setting is 'on'.

defautonuke on|**off**

Same as the **autonuke** command except that the default setting for new displays is changed. Initial setting is 'off'. Note that you can use the special 'AN' terminal capability if you want to have a dependency on the terminal type.

defbce on|**off**

Same as the **bce** command except that the default setting for new windows is changed. Initial setting is 'off'.

defbreaktype [*tcsendbreak*|*TIOCSBRK* |*TCSBRK*]

Choose one of the available methods of generating a break signal for terminal devices. The preferred methods are *tcsendbreak* and *TIOCSBRK*. The third, *TCSBRK*, blocks the complete *screen* session for the duration of the break, but it may be the only way to generate long breaks. *Tcsendbreak* and *TIOCSBRK* may or

may not produce long breaks with spikes (e.g. 4 per second). This is not only system dependant, this also differs between serial board drivers. Calling ‘‘defbreaktype’’ with no parameter displays the current setting.

defcharset [*set*]

Like the **charset** command except that the default setting for new windows is changed. Shows current default if called without argument.

defescape *xy*

Set the default command characters. This is equivalent to the ‘‘escape’’ except that it is useful multiuser sessions only. In a multiuser session ‘‘escape’’ changes the command character of the calling user, where ‘‘defescape’’ changes the default command characters for users that will be added later.

defflow on|**off**|**auto** [**interrupt**]

Same as the **flow** command except that the default setting for new windows is changed. Initial setting is ‘auto’. Specifying ‘‘defflow auto interrupt’’ is the same as the command-line options **−fa** and **−i**.

defgr on|**off**

Same as the **gr** command except that the default setting for new windows is changed. Initial setting is ‘off’.

defhstatus [*status*]

The hardstatus line that all new windows will get is set to *status*. This command is useful to make the hardstatus of every window display the window number or title or the like. *Status* may contain the same directives as in the window messages, but the directive escape character is ’^E’ (octal 005) instead of ’%’. This was done to make a misinterpretation of program generated hardstatus lines impossible. If the parameter *status* is omitted, the current default string is displayed. Per default the hardstatus line of new windows is empty.

defencoding *enc*

Same as the **encoding** command except that the default setting for new windows is changed. Initial setting is the encoding taken from the terminal.

deflog on|**off**

Same as the **log** command except that the default setting for new windows is changed. Initial setting is ‘off’.

deflogin on|**off**

Same as the **login** command except that the default setting for new windows is changed. This is initialized with ‘on’ as distributed (see config.h.in).

defmode *mode*

The mode of each newly allocated pseudo-tty is set to *mode*. *Mode* is an octal number. When no ‘‘defmode’’ command is given, mode 0622 is used.

defmonitor on|**off**

Same as the **monitor** command except that the default setting for new windows is changed. Initial setting is ‘off’.

defnonblock on|off|*numsecs*

Same as the **nonblock** command except that the default setting for displays is changed. Initial setting is ‘off’.

defobuflimit *limit*

Same as the **obuflimit** command except that the default setting for new displays is changed. Initial setting is 256 bytes. Note that you can use the special ’OL’ terminal capability if you want to have a dependency on the terminal type.

defscrollback *num*

Same as the **scrollback** command except that the default setting for new windows is changed. Initial setting is 100.

defshell *command*

Synonym to the **shell** command. See there.

defsilence on|off

Same as the **silence** command except that the default setting for new windows is changed. Initial setting is ‘off’.

defslowpaste *msec"*

Same as the **slowpaste** command except that the default setting for new windows is changed. Initial setting is 0 milliseconds, meaning ‘off’.

defutf8 on|off

Same as the **utf8** command except that the default setting for new windows is changed. Initial setting is ‘on’ if screen was started with ‘‘-U’’, otherwise ‘off’.

defwrap on|off

Same as the **wrap** command except that the default setting for new windows is changed. Initially line-wrap is on and can be toggled with the ‘‘wrap’’ command (‘‘C-a r’’) or by means of "C-a : wrap on|off".

defwritelock on|off|auto

Same as the **writelock** command except that the default setting for new windows is changed. Initially write-locks will off.

defzombie [*keys*]

Synonym to the **zombie** command. Both currently change the default. See there.

detach [**-h**]

Detach the *screen* session (disconnect it from the terminal and put it into the background). This returns you to the shell where you invoked *screen*. A detached *screen* can be resumed by invoking *screen* with the **−r** option (see also section ‘‘COMMAND-LINE OPTIONS’’). The **−h** option tells screen to immediately close the connection to the terminal (‘‘hangup’’).

dinfo

Show what screen thinks about your terminal. Useful if you want to know why features like color or the alternate charset don’t work.

displays

Shows a tabular listing of all currently connected user front-ends (displays). This is most useful for multiuser sessions.

digraph [*preset*]

This command prompts the user for a digraph sequence. The next two characters typed are looked up in a builtin table and the resulting character is inserted in the input stream. For example, if the user enters 'a"', an a-umlaut will be inserted. If the first character entered is a 0 (zero), *screen* will treat the following characters (up to three) as an octal number instead. The optional argument *preset* is treated as user input, thus one can create an ''umlaut'' key. For example the command "bindkey ^K digraph '"'" enables the user to generate an a-umlaut by typing CTRL-K a.

dumptermcap

Write the termcap entry for the virtual terminal optimized for the currently active window to the file ''.termcap'' in the user's ''$HOME/.screen'' directory (or wherever *screen* stores its sockets. See the ''FILES'' section below). This termcap entry is identical to the value of the environment variable $TERMCAP that is set up by *screen* for each window. For terminfo based systems you will need to run a converter like *captoinfo* and then compile the entry with *tic*.

echo [**-n**] *message*

The echo command may be used to annoy *screen* users with a 'message of the day'. Typically installed in a global /local/etc/screenrc. The option ''-n'' may be used to suppress the line feed. See also ''sleep''. Echo is also useful for online checking of environment variables.

encoding *enc* [*enc*]

Tell *screen* how to interpret the input/output. The first argument sets the encoding of the current window. Each window can emulate a different encoding. The optional second parameter overwrites the encoding of the connected terminal. It should never be needed as screen uses the locale setting to detect the encoding. There is also a way to select a terminal encoding depending on the terminal type by using the ''KJ'' termcap entry.

Supported encodings are eucJP, SJIS, eucKR, eucCN, Big5, GBK, KOI8-R, CP1251, UTF-8, ISO8859-2, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6, ISO8859-7, ISO8859-8, ISO8859-9, ISO8859-10, ISO8859-15, jis.

See also ''defencoding'', which changes the default setting of a new window.

escape *xy*

Set the command character to *x* and the character generating a literal command character (by triggering the ''meta'' command) to *y* (similar to the −e option). Each argument is either a single character, a two-character sequence of the form ''^x'' (meaning ''C-x''), a backslash followed by an octal number (specifying the ASCII code of the character), or a backslash followed by a second character, such as ''\^'' or ''\\''. The default is ''^Aa''.

eval *command1* [*command2* ...]

Parses and executes each argument as separate command.

exec [[*fdpat*] *newcommand* [*args* ...]]

Run a unix subprocess (specified by an executable path *newcommand* and its optional arguments) in the current window. The flow of data between newcommands stdin/stdout/stderr, the process originally started in the window (let us call it "application-process") and screen itself (window) is controlled by the

filedescriptor pattern fdpat. This pattern is basically a three character sequence representing stdin, stdout and stderr of newcommand. A dot (.) connects the file descriptor to *screen*. An exclamation mark (!) causes the file descriptor to be connected to the application-process. A colon (:) combines both. User input will go to newcommand unless newcommand receives the application-process' output (fdpats first character is '!' or ':') or a pipe symbol (|) is added (as a fourth character) to the end of fdpat.
Invoking 'exec' without arguments shows name and arguments of the currently running subprocess in this window. Only one subprocess a time can be running in each window.
When a subprocess is running the 'kill' command will affect it instead of the windows process.
Refer to the postscript file 'doc/fdpat.ps' for a confusing illustration of all 21 possible combinations. Each drawing shows the digits 2,1,0 representing the three file descriptors of newcommand. The box marked 'W' is the usual pty that has the application-process on its slave side. The box marked 'P' is the secondary pty that now has *screen* at its master side.

Abbreviations:
Whitespace between the word 'exec' and fdpat and the command can be omitted. Trailing dots and a fdpat consisting only of dots can be omitted. A simple '|' is synonymous for the pattern '!..|'; the word exec can be omitted here and can always be replaced by '!'.

Examples:

exec ... /bin/sh
exec /bin/sh
!/bin/sh

Creates another shell in the same window, while the original shell is still running. Output of both shells is displayed and user input is sent to the new /bin/sh.

exec !.. stty 19200
exec ! stty 19200
!!stty 19200

Set the speed of the window's tty. If your stty command operates on stdout, then add another '!'.

exec !..| less
|less

This adds a pager to the window output. The special character '|' is needed to give the user control over the pager although it gets its input from the window's process. This works, because *less* listens on stderr (a behavior that *screen* would not expect without the '|') when its stdin is not a tty. *Less* versions newer than 177 fail miserably here; good old *pg* still works.

!:sed -n s/.*Error.*/\007/p

Sends window output to both, the user and the sed command. The sed inserts an additional bell character (oct. 007) to the window output seen by *screen*. This will cause "Bell in window x" messages, whenever the string "Error" appears in the window.

fit

Change the window size to the size of the current region. This command is needed because screen doesn't adapt the window size automatically if the window is displayed more than once.

flow [**on**|**off**|**auto**]

Sets the flow-control mode for this window. Without parameters it cycles the current window's flow-control setting from "automatic" to "on" to "off". See the discussion on ''FLOW-CONTROL'' later on in this document for full details and note, that this is subject to change in future releases. Default is set by 'defflow'.

focus [**up**|**down**|**top**|**bottom**]

Move the input focus to the next region. This is done in a cyclic way so that the top region is selected after the bottom one. If no subcommand is given it defaults to ‘down’. ‘up’ cycles in the opposite order, ‘top’ and ‘bottom’ go to the top and bottom region respectively. Useful bindings are (j and k as in vi)

```
bind j focus down
bind k focus up
bind t focus top
bind b focus bottom
```

gr [**on**|**off**]

Turn GR charset switching on/off. Whenever screen sees an input character with the 8th bit set, it will use the charset stored in the GR slot and print the character with the 8th bit stripped. The default (see also ‘‘defgr’’) is not to process GR switching because otherwise the ISO88591 charset would not work.

hardcopy [**-h**] [*file*]

Writes out the currently displayed image to the file *file*, or, if no filename is specified, to *hardcopy.n* in the default directory, where *n* is the number of the current window. This either appends or overwrites the file if it exists. See below. If the option **-h** is specified, dump also the contents of the scrollback buffer.

hardcopy_append on|**off**

If set to "on", *screen* will append to the "hardcopy.n" files created by the command ‘‘C-a h’’, otherwise these files are overwritten each time. Default is ‘off’.

hardcopydir *directory*

Defines a directory where hardcopy files will be placed. If unset, hardcopys are dumped in *screen*’s current working directory.

hardstatus [**on**|**off**]
hardstatus [**always**]**lastline**|**message**|**ignore** [*string*]
hardstatus string [*string*]

This command configures the use and emulation of the terminal’s hardstatus line. The first form toggles whether *screen* will use the hardware status line to display messages. If the flag is set to ‘off’, these messages are overlaid in reverse video mode at the display line. The default setting is ‘on’.

The second form tells *screen* what to do if the terminal doesn’t have a hardstatus line (i.e. the termcap/terminfo capabilities "hs", "ts", "fs" and "ds" are not set). If the type ‘‘lastline’’ is used, *screen* will reserve the last line of the display for the hardstatus. ‘‘message’’ uses *screen’s* message mechanism and ‘‘ignore’’ tells *screen* never to display the hardstatus. If you prepend the word ‘‘always’’ to the type (e.g., ‘‘alwayslastline’’), *screen* will use the type even if the terminal supports a hardstatus.

The third form specifies the contents of the hardstatus line. ’%h’ is used as default string, i.e. the stored hardstatus of the current window (settable via ‘‘ESC]0;<string>^G’’ or ‘‘ESC_<string>ESC\’’) is displayed. You can customize this to any string you like including the escapes from the ‘‘STRING ESCAPES’’ chapter. If you leave out the argument *string*, the current string is displayed.

You can mix the second and third form by providing the string as additional argument.

height [**-w**|**-d**] [*lines* [*cols*]]

Set the display height to a specified number of lines. When no argument is given it toggles between 24 and 42 lines display. You can also specify a width if you want to change both values. The **-w** option tells screen to leave the display size unchanged and just set the window size, **-d** vice versa.

help [**-c** *class*]

Not really a online help, but displays a help *screen* showing you all the key bindings. The first pages list all the internal commands followed by their current bindings. Subsequent pages will display the custom commands, one command per key. Press space when you're done reading each page, or return to exit early. All other characters are ignored. If the ''-c'' option is given, display all bound commands for the specified command class. See also ''DEFAULT KEY BINDINGS'' section.

history

Usually users work with a shell that allows easy access to previous commands. For example csh has the command ''!!'' to repeat the last command executed. *Screen* allows you to have a primitive way of re-calling ''the command that started ...'': You just type the first letter of that command, then hit 'C-a {' and *screen* tries to find a previous line that matches with the 'prompt character' to the left of the cursor. This line is pasted into this window's input queue. Thus you have a crude command history (made up by the visible window and its scrollback buffer).

hstatus *status*

Change the window's hardstatus line to the string *status*.

idle [*timeout* [*cmd args*]]

Sets a command that is run after the specified number of seconds inactivity is reached. This command will normally be the ''blanker'' command to create a screen blanker, but it can be any screen command. If no command is specified, only the timeout is set. A timeout of zero (ot the special timeout **off**) disables the timer. If no arguments are given, the current settings are displayed.

ignorecase [**on**|**off**]

Tell screen to ignore the case of characters in searches. Default is 'off'.

info

Uses the message line to display some information about the current window: the cursor position in the form ''(column,row)'' starting with ''(1,1)'', the terminal width and height plus the size of the scrollback buffer in lines, like in ''(80,24)+50'', the current state of window XON/XOFF flow control is shown like this (See also section FLOW CONTROL):

+flow automatic flow control, currently on.
-flow automatic flow control, currently off.
+(+)flow flow control enabled. Agrees with automatic control.
-(+)flow flow control disabled. Disagrees with automatic control.
+(-)flow flow control enabled. Disagrees with automatic control.
-(-)flow flow control disabled. Agrees with automatic control.

The current line wrap setting ('+wrap' indicates enabled, '−wrap' not) is also shown. The flags 'ins', 'org', 'app', 'log', 'mon' or 'nored' are displayed when the window is in insert mode, origin mode, application-keypad mode, has output logging, activity monitoring or partial redraw enabled.

The currently active character set (*G0*, *G1*, *G2*, or *G3*) and in square brackets the terminal character sets that are currently designated as *G0* through *G3* is shown. If the window is in UTF-8 mode, the string ''UTF-8'' is shown instead.

Additional modes depending on the type of the window are displayed at the end of the status line (See also chapter ''WINDOW TYPES'').
If the state machine of the terminal emulator is in a non-default state, the info line is started with a string identifying the current state.

For system information use the ‘‘time’’ command.

ins_reg [*key*]

No longer exists, use ‘‘paste’’ instead.

kill

Kill current window.
If there is an ‘exec’ command running then it is killed. Otherwise the process (shell) running in the window receives a HANGUP condition, the window structure is removed and *screen* (your display) switches to another window. When the last window is destroyed, *screen* exits. After a kill *screen* switches to the previously displayed window.
Note: *Emacs* users should keep this command in mind, when killing a line. It is recommended not to use ‘‘C-a’’ as the *screen* escape key or to rebind kill to ‘‘C-a K’’.

lastmsg

Redisplay the last contents of the message/status line. Useful if you’re typing when a message appears, because the message goes away when you press a key (unless your terminal has a hardware status line). Refer to the commands ‘‘msgwait’’ and ‘‘msgminwait’’ for fine tuning.

license

Display the disclaimer page. This is done whenever *screen* is started without options, which should be often enough. See also the ‘‘startup_message’’ command.

lockscreen

Lock this display. Call a screenlock program (/local/bin/lck or /usr/bin/lock or a builtin if no other is available). Screen does not accept any command keys until this program terminates. Meanwhile processes in the windows may continue, as the windows are in the ‘detached’ state. The screenlock program may be changed through the environment variable $LOCKPRG (which must be set in the shell from which *screen* is started) and is executed with the user’s uid and gid.
Warning: When you leave other shells unlocked and you have no password set on *screen*, the lock is void: One could easily re-attach from an unlocked shell. This feature should rather be called ‘lockterminal’.

log [**on**|**off**]

Start/stop writing output of the current window to a file ‘‘screenlog.*n*’’ in the window’s default directory, where *n* is the number of the current window. This filename can be changed with the ‘logfile’ command. If no parameter is given, the state of logging is toggled. The session log is appended to the previous contents of the file if it already exists. The current contents and the contents of the scrollback history are not included in the session log. Default is ‘off’.

logfile *filename*
logfile flush *secs*

Defines the name the logfiles will get. The default is ‘‘screenlog.%n’’. The second form changes the number of seconds *screen* will wait before flushing the logfile buffer to the file-system. The default value is 10 seconds.

login [**on**|**off**]

Adds or removes the entry in the utmp database file for the current window. This controls if the window is ‘logged in’. When no parameter is given, the login state of the window is toggled. Additionally to that toggle, it is convenient having a ‘log in’ and a ‘log out’ key. E. g. ‘bind I login on’ and ‘bind O login off’ will map these keys to be C-a I and C-a O. The default setting (in config.h.in) should be ‘‘on’’ for a *screen* that runs under suid-root. Use the ‘‘deflogin’’ command to change the default login state for new windows.

Both commands are only present when *screen* has been compiled with utmp support.

logtstamp [**on**|**off**]
logtstamp after [*secs*]
logtstamp string [*string*]

This command controls logfile time-stamp mechanism of *screen.* If time-stamps are turned ‘‘on’’, *screen* adds a string containing the current time to the logfile after two minutes of inactivity. When output continues and more than another two minutes have passed, a second time-stamp is added to document the restart of the output. You can change this timeout with the second form of the command. The third form is used for customizing the time-stamp string (‘-- %n:%t -- time-stamp -- %M/%d/%y %c:%s --\n’ by default).

mapdefault

Tell *screen* that the next input character should only be looked up in the default bindkey table. See also ‘‘bindkey’’.

mapnotnext

Like mapdefault, but don’t even look in the default bindkey table.

maptimeout [*timo*]

Set the inter-character timer for input sequence detection to a timeout of *timo* ms. The default timeout is 300ms. Maptimeout with no arguments shows the current setting. See also ‘‘bindkey’’.

markkeys *string*

This is a method of changing the keymap used for copy/history mode. The string is made up of *oldchar=newchar* pairs which are separated by ‘:’. Example: The string ‘‘B=^B:F=^F’’ will change the keys ‘C-b’ and ‘C-f’ to the vi style binding (scroll up/down fill page). This happens to be the default binding for ‘B’ and ‘F’. The command ‘‘markkeys h=^B:l=^F:$=^E’’ would set the mode for an emacs-style binding. If your terminal sends characters, that cause you to abort copy mode, then this command may help by binding these characters to do nothing. The no-op character is ‘@’ and is used like this: ‘‘markkeys @=L=H’’ if you do not want to use the ‘H’ or ‘L’ commands any longer. As shown in this example, multiple keys can be assigned to one function in a single statement.

maxwin *num*

Set the maximum window number screen will create. Doesn’t affect already existing windows. The number may only be decreased.

meta

Insert the command character (C-a) in the current window’s input stream.

monitor [**on**|**off**]

Toggles activity monitoring of windows. When monitoring is turned on and an affected window is switched into the background, you will receive the activity notification message in the status line at the first sign of output and the window will also be marked with an ‘@’ in the window-status display. Monitoring is initially off for all windows.

msgminwait *sec*

Defines the time *screen* delays a new message when one message is currently displayed. The default is 1 second.

msgwait *sec*

Defines the time a message is displayed if *screen* is not disturbed by other activity. The default is 5 seconds.

multiuser on|off

Switch between singleuser and multiuser mode. Standard *screen* operation is singleuser. In multiuser mode the commands 'acladd', 'aclchg', 'aclgrp' and 'acldel' can be used to enable (and disable) other users accessing this *screen* session.

nethack on|off

Changes the kind of error messages used by *screen*. When you are familiar with the game ''nethack'', you may enjoy the nethack-style messages which will often blur the facts a little, but are much funnier to read. Anyway, standard messages often tend to be unclear as well.
This option is only available if *screen* was compiled with the NETHACK flag defined. The default setting is then determined by the presence of the environment variable $NETHACKOPTIONS.

next

Switch to the next window. This command can be used repeatedly to cycle through the list of windows.

nonblock [**on**|**off**|*numsecs*]

Tell screen how to deal with user interfaces (displays) that cease to accept output. This can happen if a user presses ^S or a TCP/modem connection gets cut but no hangup is received. If nonblock is **off** (this is the default) screen waits until the display restarts to accept the output. If nonblock is **on**, screen waits until the timeout is reached (**on** is treated as 1s). If the display still doesn't receive characters, screen will consider it ''blocked'' and stop sending characters to it. If at some time it restarts to accept characters, screen will unblock the display and redisplay the updated window contents.

number [*n*]

Change the current windows number. If the given number *n* is already used by another window, both windows exchange their numbers. If no argument is specified, the current window number (and title) is shown.

obuflimit [*limit*]

If the output buffer contains more bytes than the specified limit, no more data will be read from the windows. The default value is 256. If you have a fast display (like xterm), you can set it to some higher value. If no argument is specified, the current setting is displayed.

only

Kill all regions but the current one.

other

Switch to the window displayed previously. If this window does no longer exist, *other* has the same effect as *next*.

partial on|off

Defines whether the display should be refreshed (as with *redisplay*) after switching to the current window. This command only affects the current window. To immediately affect all windows use the *allpartial* command. Default is 'off', of course. This default is fixed, as there is currently no *defpartial* command.

password [*crypted_pw*]

Present a crypted password in your ''.screenrc'' file and *screen* will ask for it, whenever someone attempts to resume a detached. This is useful if you have privileged programs running under *screen* and you want to

protect your session from reattach attempts by another user masquerading as your uid (i.e. any superuser.) If no crypted password is specified, *screen* prompts twice for typing a password and places its encryption in the paste buffer. Default is 'none', this disables password checking.

paste [*registers* [*dest_reg*]]

Write the (concatenated) contents of the specified registers to the stdin queue of the current window. The register '.' is treated as the paste buffer. If no parameter is given the user is prompted for a single register to paste. The paste buffer can be filled with the *copy*, *history* and *readbuf* commands. Other registers can be filled with the *register*, *readreg* and *paste* commands. If *paste* is called with a second argument, the contents of the specified registers is pasted into the named destination register rather than the window. If '.' is used as the second argument, the displays paste buffer is the destination. Note, that ''paste'' uses a wide variety of resources: Whenever a second argument is specified no current window is needed. When the source specification only contains registers (not the paste buffer) then there need not be a current display (terminal attached), as the registers are a global resource. The paste buffer exists once for every user.

pastefont [**on**|**off**]

Tell *screen* to include font information in the paste buffer. The default is not to do so. This command is especially useful for multi character fonts like kanji.

pow_break

Reopen the window's terminal line and send a break condition. See 'break'.

pow_detach

Power detach. Mainly the same as *detach*, but also sends a HANGUP signal to the parent process of *screen*. CAUTION: This will result in a logout, when *screen* was started from your login shell.

pow_detach_msg [*message*]

The *message* specified here is output whenever a 'Power detach' was performed. It may be used as a replacement for a logout message or to reset baud rate, etc. Without parameter, the current message is shown.

prev

Switch to the window with the next lower number. This command can be used repeatedly to cycle through the list of windows.

printcmd [*cmd*]

If *cmd* is not an empty string, *screen* will not use the terminal capabilities ''po/pf'' if it detects an ansi print sequence **ESC [5 i**, but pipe the output into *cmd*. This should normally be a command like ''lpr'' or ''cat > /tmp/scrprint''. **printcmd** without a command displays the current setting. The ansi sequence **ESC ** ends printing and closes the pipe.
Warning: Be careful with this command! If other user have write access to your terminal, they will be able to fire off print commands.

process [*key*]

Stuff the contents of the specified register into *screen*'s input queue. If no argument is given you are prompted for a register name. The text is parsed as if it had been typed in from the user's keyboard. This command can be used to bind multiple actions to a single key.

quit

Kill all windows and terminate *screen*. Note that on VT100-style terminals the keys C-4 and C-\ are identical. This makes the default bindings dangerous: Be careful not to type C-a C-4 when selecting window no. 4. Use the empty bind command (as in ‘‘bind ’^\’’) to remove a key binding.

readbuf [**-e** *encoding*] [*filename*]

Reads the contents of the specified file into the paste buffer. You can tell screen the encoding of the file via the **-e** option. If no file is specified, the screen-exchange filename is used. See also ‘‘bufferfile’’ command.

readreg [**-e** *encoding*] [*register* [*filename*]]

Does one of two things, dependent on number of arguments: with zero or one arguments it it duplicates the paste buffer contents into the register specified or entered at the prompt. With two arguments it reads the contents of the named file into the register, just as *readbuf* reads the screen-exchange file into the paste buffer. You can tell screen the encoding of the file via the **-e** option. The following example will paste the system’s password file into the *screen* window (using register p, where a copy remains):

C-a : readreg p /etc/passwd
C-a : paste p

redisplay

Redisplay the current window. Needed to get a full redisplay when in partial redraw mode.

register [**-e** *encoding*] *key string*

Save the specified *string* to the register *key*. The encoding of the string can be specified via the **-e** option. See also the ‘‘paste’’ command.

remove

Kill the current region. This is a no-op if there is only one region.

removebuf

Unlinks the screen-exchange file used by the commands ‘‘writebuf’’ and ‘‘readbuf’’.

reset

Reset the virtual terminal to its ‘‘power-on’’ values. Useful when strange settings (like scroll regions or graphics character set) are left over from an application.

resize

Resize the current region. The space will be removed from or added to the region below or if there’s not enough space from the region above.

resize +N	increase current region height by N
resize -N	decrease current region height by N
resize N	set current region height to N
resize =	make all windows equally high
resize max	maximize current region height
resize min	minimize current region height

screen [*-opts*] [*n*] [*cmd* [*args*]]

Establish a new window. The flow-control options (**−f**, **−fn** and **−fa**), title (a. k. a.) option (**−t**), login options (**-l** and **-ln**) , terminal type option (**-T** <term>), the all-capability-flag (**-a**) and scrollback option (**-h** <num>) may be specified with each command. The option (**-M**) turns monitoring on for this window. The option (**-L**) turns output logging on for this window. If an optional number *n* in the range 0..9 is given, the window number *n* is assigned to the newly created window (or, if this number is already in-use, the next available number). If a command is specified after ''screen'', this command (with the given arguments) is started in the window; otherwise, a shell is created. Thus, if your ''.screenrc'' contains the lines

```
# example for .screenrc:
screen 1
screen -fn -t foobar -L 2 telnet foobar
```

screen creates a shell window (in window #1) and a window with a TELNET connection to the machine foobar (with no flow-control using the title ''foobar'' in window #2) and will write a logfile (''screenlog.2'') of the telnet session. Note, that unlike previous versions of *screen* no additional default window is created when ''screen'' commands are included in your ''.screenrc'' file. When the initialization is completed, *screen* switches to the last window specified in your .screenrc file or, if none, opens a default window #0. Screen has built in some functionality of ''cu'' and ''telnet''. See also chapter ''WINDOW TYPES''.

scrollback *num*

Set the size of the scrollback buffer for the current windows to *num* lines. The default scrollback is 100 lines. See also the ''defscrollback'' command and use ''C-a i'' to view the current setting.

select [*WindowID*]

Switch to the window identified by *WindowID*. This can be a prefix of a window title (alphanumeric window name) or a window number. The parameter is optional and if omitted, you get prompted for an identifier. When a new window is established, the first available number is assigned to this window. Thus, the first window can be activated by ''select 0''. The number of windows is limited at compile-time by the MAXWIN configuration parameter. There are two special WindowIDs, ''-'' selects the internal blank window and ''.'' selects the current window. The latter is useful if used with screen's ''-X'' option.

sessionname [*name*]

Rename the current session. Note, that for ''screen -list'' the name shows up with the process-id prepended. If the argument ''name'' is omitted, the name of this session is displayed. Caution: The $STY environment variables still reflects the old name. This may result in confusion. The default is constructed from the tty and host names.

setenv [*var* [*string*]]

Set the environment variable *var* to value *string*. If only *var* is specified, the user will be prompted to enter a value. If no parameters are specified, the user will be prompted for both variable and value. The environment is inherited by all subsequently forked shells.

setsid [**on**|**off**]

Normally screen uses different sessions and process groups for the windows. If setsid is turned *off*, this is not done anymore and all windows will be in the same process group as the screen backend process. This also breaks job-control, so be careful. The default is *on*, of course. This command is probably useful only in rare circumstances.

shell *command*

Set the command to be used to create a new shell. This overrides the value of the environment variable $SHELL. This is useful if you'd like to run a tty-enhancer which is expecting to execute the program specified in $SHELL. If the command begins with a '-' character, the shell will be started as a login-shell.

shelltitle *title*

Set the title for all shells created during startup or by the C-A C-c command. For details about what a title is, see the discussion entitled ''TITLES (naming windows)''.

silence [**on**|**off**|*sec*]

Toggles silence monitoring of windows. When silence is turned on and an affected window is switched into the background, you will receive the silence notification message in the status line after a specified period of inactivity (silence). The default timeout can be changed with the 'silencewait' command or by specifying a number of seconds instead of 'on' or 'off'. Silence is initially off for all windows.

silencewait *sec*

Define the time that all windows monitored for silence should wait before displaying a message. Default 30 seconds.

sleep *num*

This command will pause the execution of a .screenrc file for *num* seconds. Keyboard activity will end the sleep. It may be used to give users a chance to read the messages output by ''echo''.

slowpaste *msec*

Define the speed at which text is inserted into the current window by the paste ("C-a]") command. If the slowpaste value is nonzero text is written character by character. *screen* will make a pause of *msec* milliseconds after each single character write to allow the application to process its input. Only use slowpaste if your underlying system exposes flow control problems while pasting large amounts of text.

source *file*

Read and execute commands from file *file*. Source commands may be nested to a maximum recursion level of ten. If file is not an absolute path and screen is already processing a source command, the parent directory of the running source command file is used to search for the new command file before screen's current directory.

Note that termcap/terminfo/termcapinfo commands only work at startup and reattach time, so they must be reached via the default screenrc files to have an effect.

sorendition [*attr* [*color*]]

Change the way *screen* does highlighting for text marking and printing messages. See the ''STRING ESCAPES'' chapter for the syntax of the modifiers. The default is currently ''=s dd'' (standout, default colors).

split

Split the current region into two new ones. All regions on the display are resized to make room for the new region. The blank window is displayed on the new region. Use the ''remove'' or the ''only'' command to delete regions.

startup_message on|off

Select whether you want to see the copyright notice during startup. Default is ‘on’, as you probably noticed.

stuff *string*

Stuff the string *string* in the input buffer of the current window. This is like the ‘‘paste’’ command but with much less overhead. You cannot paste large buffers with the ‘‘stuff’’ command. It is most useful for key bindings. See also ‘‘bindkey’’.

su [**username** [**password** [**password2**]]

Substitute the user of a display. The command prompts for all parameters that are omitted. If passwords are specified as parameters, they have to be specified un-crypted. The first password is matched against the systems passwd database, the second password is matched against the *screen* password as set with the commands ‘‘acladd’’ or ‘‘password’’. ‘‘Su’’ may be useful for the *screen* administrator to test multiuser setups. When the identification fails, the user has access to the commands available for user **nobody**. These are ‘‘detach’’, ‘‘license’’, ‘‘version’’, ‘‘help’’ and ‘‘displays’’.

suspend

Suspend *screen*. The windows are in the ‘detached’ state, while *screen* is suspended. This feature relies on the shell being able to do job control.

term *term*

In each window’s environment *screen* opens, the $TERM variable is set to ‘‘screen’’ by default. But when no description for ‘‘screen’’ is installed in the local termcap or terminfo data base, you set $TERM to − say − ‘‘vt100’’. This won’t do much harm, as *screen* is VT100/ANSI compatible. The use of the ‘‘term’’ command is discouraged for non-default purpose. That is, one may want to specify special $TERM settings (e.g. vt100) for the next ‘‘screen rlogin othermachine’’ command. Use the command ‘‘screen -T vt100 rlogin othermachine’’ rather than setting and resetting the default.

termcap *term terminal-tweaks* [*window-tweaks*]
terminfo *term terminal-tweaks* [*window-tweaks*]
termcapinfo *term terminal-tweaks* [*window-tweaks*]

Use this command to modify your terminal’s termcap entry without going through all the hassles involved in creating a custom termcap entry. Plus, you can optionally customize the termcap generated for the windows. You have to place these commands in one of the screenrc startup files, as they are meaningless once the terminal emulator is booted.
If your system works uses the terminfo database rather than termcap, *screen* will understand the ‘terminfo’ command, which has the same effects as the ‘termcap’ command. Two separate commands are provided, as there are subtle syntactic differences, e.g. when parameter interpolation (using ‘%’) is required. Note that termcap names of the capabilities have to be used with the ‘terminfo’ command.
In many cases, where the arguments are valid in both terminfo and termcap syntax, you can use the command ‘termcapinfo’, which is just a shorthand for a pair of ‘termcap’ and ‘terminfo’ commands with identical arguments.

The first argument specifies which terminal(s) should be affected by this definition. You can specify multiple terminal names by separating them with ‘|’s. Use ‘*’ to match all terminals and ‘vt*’ to match all terminals that begin with ‘‘vt’’.

Each *tweak* argument contains one or more termcap defines (separated by ‘:’s) to be inserted at the start of the appropriate termcap entry, enhancing it or overriding existing values. The first tweak modifies your terminal’s termcap, and contains definitions that your terminal uses to perform certain functions. Specify a null string to leave this unchanged (e. g. ’’). The second (optional) tweak modifies all the window termcaps, and should contain definitions that *screen* understands (see the ‘‘VIRTUAL TERMINAL’’ section).

Some examples:

```
termcap xterm*  LP:hs@
```

Informs *screen* that all terminals that begin with ‘xterm’ have firm auto-margins that allow the last position on the screen to be updated (LP), but they don’t really have a status line (no ’hs’ − append ‘@’ to turn entries off). Note that we assume ‘LP’ for all terminal names that start with ‘‘vt’’, but only if you don’t specify a termcap command for that terminal.

```
termcap vt*  LP
termcap vt102|vt220  Z0=\E[?3h:Z1=\E[?3l
```

Specifies the firm-margined ‘LP’ capability for all terminals that begin with ‘vt’, and the second line will also add the escape-sequences to switch into (Z0) and back out of (Z1) 132-character-per-line mode if this is a VT102 or VT220. (You must specify Z0 and Z1 in your termcap to use the width-changing commands.)

```
termcap vt100  ""  l0=PF1:l1=PF2:l2=PF3:l3=PF4
```

This leaves your vt100 termcap alone and adds the function key labels to each window’s termcap entry.

```
termcap h19|z19  am@:im=\E@:ei=\EO  dc=\E[P
```

Takes a h19 or z19 termcap and turns off auto-margins (am@) and enables the insert mode (im) and end-insert (ei) capabilities (the ‘@’ in the ‘im’ string is after the ‘=’, so it is part of the string). Having the ‘im’ and ‘ei’ definitions put into your terminal’s termcap will cause *screen* to automatically advertise the character-insert capability in each window’s termcap. Each window will also get the delete-character capability (dc) added to its termcap, which *screen* will translate into a line-update for the terminal (we’re pretending it doesn’t support character deletion).

If you would like to fully specify each window’s termcap entry, you should instead set the $SCREENCAP variable prior to running *screen*. See the discussion on the ‘‘VIRTUAL TERMINAL’’ in this manual, and the termcap(5) man page for more information on termcap definitions.

time [*string*]

Uses the message line to display the time of day, the host name, and the load averages over 1, 5, and 15 minutes (if this is available on your system). For window specific information use ‘‘info’’.

If a string is specified, it changes the format of the time report like it is described in the ‘‘STRING ESCAPES’’ chapter. Screen uses a default of "%c:%s %M %d %H%? %l%?".

title [*windowtitle*]

Set the name of the current window to *windowtitle*. If no name is specified, *screen* prompts for one. This command was known as ‘aka’ in previous releases.

unsetenv *var*

Unset an environment variable.

utf8 [**on**|**off** [**on**|**off**]]

Change the encoding used in the current window. If utf8 is enabled, the strings sent to the window will be UTF-8 encoded and vice versa. Omitting the parameter toggles the setting. If a second parameter is given, the display’s encoding is also changed (this should rather be done with screen’s ‘‘-U’’ option). See also ‘‘defutf8’’, which changes the default setting of a new window.

vbell [**on**|**off**]

Sets the visual bell setting for this window. Omitting the parameter toggles the setting. If vbell is switched on, but your terminal does not support a visual bell, a ‘vbell-message’ is displayed in the status line when the bell character (^G) is received. Visual bell support of a terminal is defined by the termcap variable ‘vb’

(terminfo: 'flash').
Per default, vbell is off, thus the audible bell is used. See also 'bell_msg'.

vbell_msg [*message*]

Sets the visual bell message. *message* is printed to the status line if the window receives a bell character (^G), vbell is set to ''on'', but the terminal does not support a visual bell. The default message is ''Wuff, Wuff!!''. Without parameter, the current message is shown.

vbellwait *sec*

Define a delay in seconds after each display of *screen*'s visual bell message. The default is 1 second.

verbose [**on**|**off**]

If verbose is switched on, the command name is echoed, whenever a window is created (or resurrected from zombie state). Default is off. Without parameter, the current setting is shown.

version

Print the current version and the compile date in the status line.

wall *message*

Write a message to all displays. The message will appear in the terminal's status line.

width [**-w**|**-d**] [*cols* [*lines*]]

Toggle the window width between 80 and 132 columns or set it to *cols* columns if an argument is specified. This requires a capable terminal and the termcap entries ''Z0'' and ''Z1''. See the ''termcap'' command for more information. You can also specify a new height if you want to change both values. The **-w** option tells screen to leave the display size unchanged and just set the window size, **-d** vice versa.

windowlist [**-b**] [**-m**]
windowlist string [*string*]
windowlist title [*title*]

Display all windows in a table for visual window selection. The desired window can be selected via the standard movement keys (see the ''copy'' command) and activated via the return key. If the **-b** option is given, screen will switch to the blank window before presenting the list, so that the current window is also selectable. The **-m** option changes the order of the windows, instead of sorting by window numbers screen uses its internal most-recently-used list.

The table format can be changed with the **string** and **title** option, the title is displayed as table heading, while the lines are made by using the string setting. The default setting is ''Num Name%=Flags'' for the title and ''%3n %t%=%f'' for the lines. See the ''STRING ESCAPES'' chapter for more codes (e.g. color settings).

windows

Uses the message line to display a list of all the windows. Each window is listed by number with the name of process that has been started in the window (or its title); the current window is marked with a '*'; the previous window is marked with a '-'; all the windows that are ''logged in'' are marked with a '$'; a background window that has received a bell is marked with a '!'; a background window that is being monitored and has had activity occur is marked with an '@'; a window which has output logging turned on is marked with '(L)'; windows occupied by other users are marked with '&'; windows in the zombie state are marked with 'Z'. If this list is too long to fit on the terminal's status line only the portion around the current window is displayed.

wrap [**on**|**off**]

Sets the line-wrap setting for the current window. When line-wrap is on, the second consecutive printable character output at the last column of a line will wrap to the start of the following line. As an added feature, backspace (^H) will also wrap through the left margin to the previous line. Default is 'on'.

writebuf [**-e** *encoding*] [*filename*]

Writes the contents of the paste buffer to the specified file, or the public accessible screen-exchange file if no filename is given. This is thought of as a primitive means of communication between *screen* users on the same host. If an encoding is specified the paste buffer is recoded on the fly to match the encoding. The filename can be set with the *bufferfile* command and defaults to ''/tmp/screen-exchange''.

writelock [**on**|**off**|**auto**]

In addition to access control lists, not all users may be able to write to the same window at once. Per default, writelock is in 'auto' mode and grants exclusive input permission to the user who is the first to switch to the particular window. When he leaves the window, other users may obtain the writelock (automatically). The writelock of the current window is disabled by the command ''writelock off''. If the user issues the command ''writelock on'' he keeps the exclusive write permission while switching to other windows.

xoff
xon

Insert a CTRL-s / CTRL-q character to the stdin queue of the current window.

zmodem [**off**|**auto**|**catch**|**pass**]
zmodem sendcmd [*string*]
zmodem recvcmd [*string*]

Define zmodem support for screen. Screen understands two different modes when it detects a zmodem request: ''pass'' and ''catch''. If the mode is set to ''pass'', screen will relay all data to the attacher until the end of the transmission is reached. In ''catch'' mode screen acts as a zmodem endpoint and starts the corresponding rz/sz commands. If the mode is set to ''auto'', screen will use ''catch'' if the window is a tty (e.g. a serial line), otherwise it will use ''pass''.
You can define the templates screen uses in ''catch'' mode via the second and the third form.
Note also that this is an experimental feature.

zombie [*keys*]
defzombie [*keys*]

Per default *screen* windows are removed from the window list as soon as the windows process (e.g. shell) exits. When a string of two keys is specified to the zombie command, 'dead' windows will remain in the list. The **kill** command may be used to remove such a window. Pressing the first key in the dead window has the same effect. When pressing the second key, *screen* will attempt to resurrect the window. The process that was initially running in the window will be launched again. Calling **zombie** without parameters will clear the zombie setting, thus making windows disappear when their process exits.

As the zombie-setting is manipulated globally for all windows, this command should only be called **defzombie**. Until we need this as a per window setting, the commands **zombie** and **defzombie** are synonymous.

THE MESSAGE LINE

Screen displays informational messages and other diagnostics in a *message line*. While this line is distributed to appear at the bottom of the screen, it can be defined to appear at the top of the screen during compilation. If your terminal has a status line defined in its termcap, *screen* will use this for displaying its messages, otherwise a line of the current screen will be temporarily overwritten and output will be

momentarily interrupted. The message line is automatically removed after a few seconds delay, but it can also be removed early (on terminals without a status line) by beginning to type.

The message line facility can be used by an application running in the current window by means of the ANSI *Privacy message* control sequence. For instance, from within the shell, try something like:

echo '<esc>^Hello world from window '$WINDOW'<esc>\\'

where '<esc>' is an *escape*, '^' is a literal up-arrow, and '\\' turns into a single backslash.

WINDOW TYPES

Screen provides three different window types. New windows are created with *screen*'s **screen** command (see also the entry in chapter ''CUSTOMIZATION''). The first parameter to the **screen** command defines which type of window is created. The different window types are all special cases of the normal type. They have been added in order to allow *screen* to be used efficiently as a console multiplexer with 100 or more windows.

- The normal window contains a shell (default, if no parameter is given) or any other system command that could be executed from a shell (e.g. **slogin**, etc...)

- If a tty (character special device) name (e.g. ''/dev/ttya'') is specified as the first parameter, then the window is directly connected to this device. This window type is similar to ''screen cu -l /dev/ttya''. Read and write access is required on the device node, an exclusive open is attempted on the node to mark the connection line as busy. An optional parameter is allowed consisting of a comma separated list of flags in the notation used by stty(1):

  <baud_rate>
  Usually 300, 1200, 9600 or 19200. This affects transmission as well as receive speed.

  cs8 or cs7
  Specify the transmission of eight (or seven) bits per byte.

  ixon or -ixon
  Enables (or disables) software flow-control (CTRL-S/CTRL-Q) for sending data.

  ixoff or -ixon
  Enables (or disables) software flow-control for receiving data.

  istrip or -istrip
  Clear (or keep) the eight bit in each received byte.

  You may want to specify as many of these options as applicable. Unspecified options cause the terminal driver to make up the parameter values of the connection. These values are system dependant and may be in defaults or values saved from a previous connection.

  For tty windows, the **info** command shows some of the modem control lines in the status line. These may include 'RTS', 'CTS', 'DTR', 'DSR', 'CD' and more. This depends on the available ioctl()'s and system header files as well as the on the physical capabilities of the serial board. Signals that are logical low (inactive) have their name preceded by an exclamation mark (!), otherwise the signal is logical high (active). Signals not supported by the hardware but available to the ioctl() interface are usually shown low.
  When the CLOCAL status bit is true, the whole set of modem signals is placed inside curly braces ({ and }). When the CRTSCTS or TIOCSOFTCAR bit is set, the signals 'CTS' or 'CD' are shown in parenthesis, respectively.

  For tty windows, the command **break** causes the Data transmission line (TxD) to go low for a specified period of time. This is expected to be interpreted as break signal on the other side. No data is sent and no modem control line is changed when a **break** is issued.

- If the first parameter is ‘‘//telnet’’, the second parameter is expected to be a host name, and an optional third parameter may specify a TCP port number (default decimal 23). Screen will connect to a server listening on the remote host and use the telnet protocol to communicate with that server.
  For telnet windows, the command **info** shows details about the connection in square brackets ([and]) at the end of the status line.

  b BINARY. The connection is in binary mode.

  e ECHO. Local echo is disabled.

  c SGA. The connection is in ‘character mode’ (default: ‘line mode’).

  t TTYPE. The terminal type has been requested by the remote host. Screen sends the name ‘‘screen’’ unless instructed otherwise (see also the command ‘term’).

  w NAWS. The remote site is notified about window size changes.

  f LFLOW. The remote host will send flow control information. (Ignored at the moment.)

  Additional flags for debugging are x, t and n (XDISPLOC, TSPEED and NEWENV).

  For telnet windows, the command **break** sends the telnet code IAC BREAK (decimal 243) to the remote host.

  This window type is only available if *screen* was compiled with the BUILTIN_TELNET option defined.

STRING ESCAPES

Screen provides an escape mechanism to insert information like the current time into messages or file names. The escape character is ’%’ with one exception: inside of a window’s hardstatus ’^%’ (’^E’) is used instead.

Here is the full list of supported escapes:

% the escape character itself

a either ’am’ or ’pm’

A either ’AM’ or ’PM’

c current time HH:MM in 24h format

C current time HH:MM in 12h format

d day number

D weekday name

f flags of the window

F sets %? to true if the window has the focus

h hardstatus of the window

H hostname of the system

l current load of the system

m month number

M month name

n window number

s seconds

t window title

u all other users on this window

w all window numbers and names. With '-' quailifier: up to the current window; with '+' qualifier: starting with the window after the current one.

W all window numbers and names except the current one

y last two digits of the year number

Y full year number

? the part to the next '%?' is displayed only if a '%' escape inside the part expands to a non-empty string

: else part of '%?'

= pad the string to the display's width (like TeX's hfill). If a number is specified, pad to the percentage of the window's width. A '0' qualifier tells screen to treat the number as absolute position. You can specify to pad relative to the last absolute pad position by adding a '+' qualifier or to pad relative to the right margin by using '-'. The padding truncates the string if the specified position lies before the current position. Add the 'L' qualifier to change this.

< same as '%=' but just do truncation, do not fill with spaces

> mark the current text position for the next truncation. When screen needs to do truncation, it tries to do it in a way that the marked position gets moved to the specified percentage of the output area. (The area starts from the last absolute pad position and ends with the position specified by the truncation operator.) The 'L' qualifier tells screen to mark the truncated parts with '...'.

{ attribute/color modifier string terminated by the next ''}''

‘ Substitute with the output of a 'backtick' command. The length qualifier is misused to identify one of the commands.

The 'c' and 'C' escape may be qualified with a '0' to make *screen* use zero instead of space as fill character. The '0' qualifier also makes the '=' escape use absolute positions. The 'n' and '=' escapes understand a length qualifier (e.g. '%3n'), 'D' and 'M' can be prefixed with 'L' to generate long names, 'w' and 'W' also show the window flags if 'L' is given.

An attribute/color modifier is is used to change the attributes or the color settings. Its format is ''[attribute modifier] [color description]''. The attribute modifier must be prefixed by a change type indicator if it can be confused with a color desciption. The following change types are known:

+ add the specified set to the current attributes

- remove the set from the current attributes

! invert the set in the current attributes

= change the current attributes to the specified set

The attribute set can either be specified as a hexadecimal number or a combination of the following letters:

d dim
u underline
b bold
r reverse
s standout
B blinking

Colors are coded either as a hexadecimal number or two letters specifying the desired background and foreground color (in that order). The following colors are known:

k black
r red
g green

y	yellow
b	blue
m	magenta
c	cyan
w	white
d	default color
.	leave color unchanged

The capitalized versions of the letter specify bright colors. You can also use the pseudo-color 'i' to set just the brightness and leave the color unchanged.
A one digit/letter color description is treated as foreground or background color dependant on the current attributes: if reverse mode is set, the background color is changed instead of the foreground color. If you don't like this, prefix the color with a ''.''. If you want the same behaviour for two-letter color descriptions, also prefix them with a ''.''.
As a special case, ''%{-}'' restores the attributes and colors that were set before the last change was made (i.e. pops one level of the color-change stack).

Examples:

''G'' set color to bright green

''+b r'' use bold red

''= yd'' clear all attributes, write in default color on yellow background.

%-Lw%{= BW}%50>%n%f* %t%{-}%+Lw%<
The available windows centered at the current window and truncated to the available width. The current window is displayed white on blue. This can be used with ''hardstatus alwayslastline''.

%?%F%{.R.}%?%3n %t%? [%h]%?
The window number and title and the window's hardstatus, if one is set. Also use a red background if this is the active focus. Useful for ''caption string''.

FLOW-CONTROL

Each window has a flow-control setting that determines how *screen* deals with the XON and XOFF characters (and perhaps the interrupt character). When flow-control is turned off, *screen* ignores the XON and XOFF characters, which allows the user to send them to the current program by simply typing them (useful for the *emacs* editor, for instance). The trade-off is that it will take longer for output from a ''normal'' program to pause in response to an XOFF. With flow-control turned on, XON and XOFF characters are used to immediately pause the output of the current window. You can still send these characters to the current program, but you must use the appropriate two-character *screen* commands (typically ''C-a q'' (xon) and ''C-a s'' (xoff)). The xon/xoff commands are also useful for typing C-s and C-q past a terminal that intercepts these characters.

Each window has an initial flow-control value set with either the **−f** option or the ''defflow'' .screenrc command. Per default the windows are set to automatic flow-switching. It can then be toggled between the three states 'fixed on', 'fixed off' and 'automatic' interactively with the ''flow'' command bound to "C-a f".

The automatic flow-switching mode deals with flow control using the TIOCPKT mode (like ''rlogin'' does). If the tty driver does not support TIOCPKT, *screen* tries to find out the right mode based on the current setting of the application keypad − when it is enabled, flow-control is turned off and visa versa. Of course, you can still manipulate flow-control manually when needed.

If you're running with flow-control enabled and find that pressing the interrupt key (usually C-c) does not interrupt the display until another 6-8 lines have scrolled by, try running *screen* with the ''interrupt'' option (add the ''interrupt'' flag to the ''flow'' command in your .screenrc, or use the **−i** command-line option). This causes the output that *screen* has accumulated from the interrupted program to be flushed. One disadvantage is that the virtual terminal's memory contains the non-flushed version of the output, which in rare cases can cause minor inaccuracies in the output. For example, if you switch screens and return, or update the screen with ''C-a l'' you would see the version of the output you would have gotten without ''interrupt'' being on. Also, you might need to turn off flow-control (or use auto-flow mode to turn it off automatically)

when running a program that expects you to type the interrupt character as input, as it is possible to interrupt the output of the virtual terminal to your physical terminal when flow-control is enabled. If this happens, a simple refresh of the screen with ''C-a l'' will restore it. Give each mode a try, and use whichever mode you find more comfortable.

TITLES (naming windows)

You can customize each window's name in the window display (viewed with the ''windows'' command (C-a w)) by setting it with one of the title commands. Normally the name displayed is the actual command name of the program created in the window. However, it is sometimes useful to distinguish various programs of the same name or to change the name on-the-fly to reflect the current state of the window.

The default name for all shell windows can be set with the ''shelltitle'' command in the .screenrc file, while all other windows are created with a ''screen'' command and thus can have their name set with the **−t** option. Interactively, there is the title-string escape-sequence (<esc>k*name*<esc>\) and the ''title'' command (C-a A). The former can be output from an application to control the window's name under software control, and the latter will prompt for a name when typed. You can also bind pre-defined names to keys with the ''title'' command to set things quickly without prompting.

Finally, *screen* has a shell-specific heuristic that is enabled by setting the window's name to ''*search*|*name*'' and arranging to have a null title escape-sequence output as a part of your prompt. The *search* portion specifies an end-of-prompt search string, while the *name* portion specifies the default shell name for the window. If the *name* ends in a ':' *screen* will add what it believes to be the current command running in the window to the end of the window's shell name (e. g. ''*name:cmd*''). Otherwise the current command name supersedes the shell name while it is running.

Here's how it works: you must modify your shell prompt to output a null title-escape-sequence (<esc>k<esc>\) as a part of your prompt. The last part of your prompt must be the same as the string you specified for the *search* portion of the title. Once this is set up, *screen* will use the title-escape-sequence to clear the previous command name and get ready for the next command. Then, when a newline is received from the shell, a search is made for the end of the prompt. If found, it will grab the first word after the matched string and use it as the command name. If the command name begins with either '!', '%', or '^' *screen* will use the first word on the following line (if found) in preference to the just-found name. This helps csh users get better command names when using job control or history recall commands.

Here's some .screenrc examples:

```
screen -t top 2 nice top
```

Adding this line to your .screenrc would start a nice-d version of the ''top'' command in window 2 named ''top'' rather than ''nice''.

```
shelltitle '> |csh'
screen 1
```

These commands would start a shell with the given shelltitle. The title specified is an auto-title that would expect the prompt and the typed command to look something like the following:

```
/usr/joe/src/dir> trn
```

(it looks after the '> ' for the command name). The window status would show the name ''trn'' while the command was running, and revert to ''csh'' upon completion.

```
bind R screen -t '% |root:' su
```

Having this command in your .screenrc would bind the key sequence ''C-a R'' to the ''su'' command and give it an auto-title name of ''root:''. For this auto-title to work, the screen could look something like this:

```
% !em
emacs file.c
```

Here the user typed the csh history command ‘‘!em’’ which ran the previously entered ‘‘emacs’’ command. The window status would show ‘‘root:emacs’’ during the execution of the command, and revert to simply ‘‘root:’’ at its completion.

```
bind o title
bind E title ""
bind u title (unknown)
```

The first binding doesn’t have any arguments, so it would prompt you for a title. when you type ‘‘C-a o’’. The second binding would clear an auto-title’s current setting (C-a E). The third binding would set the current window’s title to ‘‘(unknown)’’ (C-a u).

One thing to keep in mind when adding a null title-escape-sequence to your prompt is that some shells (like the csh) count all the non-control characters as part of the prompt’s length. If these invisible characters aren’t a multiple of 8 then backspacing over a tab will result in an incorrect display. One way to get around this is to use a prompt like this:

```
set prompt='^[[0000m^[k^[\% '
```

The escape-sequence ‘‘<esc>[0000m’’ not only normalizes the character attributes, but all the zeros round the length of the invisible characters up to 8. Bash users will probably want to echo the escape sequence in the PROMPT_COMMAND:

```
PROMPT_COMMAND='echo -n -e "\033k\033\134"'
```

(I used ‘‘134’’ to output a ‘\’ because of a bug in bash v1.04).

THE VIRTUAL TERMINAL

Each window in a *screen* session emulates a VT100 terminal, with some extra functions added. The VT100 emulator is hard-coded, no other terminal types can be emulated.
Usually *screen* tries to emulate as much of the VT100/ANSI standard as possible. But if your terminal lacks certain capabilities, the emulation may not be complete. In these cases *screen* has to tell the applications that some of the features are missing. This is no problem on machines using termcap, because *screen* can use the $TERMCAP variable to customize the standard *screen* termcap.

But if you do a rlogin on another machine or your machine supports only terminfo this method fails. Because of this, *screen* offers a way to deal with these cases. Here is how it works:

When *screen* tries to figure out a terminal name for itself, it first looks for an entry named ‘‘screen.<term>’’, where <term> is the contents of your $TERM variable. If no such entry exists, *screen* tries ‘‘screen’’ (or ‘‘screen-w’’ if the terminal is wide (132 cols or more)). If even this entry cannot be found, ‘‘vt100’’ is used as a substitute.

The idea is that if you have a terminal which doesn’t support an important feature (e.g. delete char or clear to EOS) you can build a new termcap/terminfo entry for *screen* (named ‘‘screen.<dumbterm>’’) in which this capability has been disabled. If this entry is installed on your machines you are able to do a rlogin and still keep the correct termcap/terminfo entry. The terminal name is put in the $TERM variable of all new windows. *Screen* also sets the $TERMCAP variable reflecting the capabilities of the virtual terminal emulated. Notice that, however, on machines using the terminfo database this variable has no effect. Furthermore, the variable $WINDOW is set to the window number of each window.

The actual set of capabilities supported by the virtual terminal depends on the capabilities supported by the physical terminal. If, for instance, the physical terminal does not support underscore mode, *screen* does not put the ‘us’ and ‘ue’ capabilities into the window’s $TERMCAP variable, accordingly. However, a minimum number of capabilities must be supported by a terminal in order to run *screen*; namely scrolling, clear screen, and direct cursor addressing (in addition, *screen* does not run on hardcopy terminals or on terminals that over-strike).

Also, you can customize the $TERMCAP value used by *screen* by using the ‘‘termcap’’ .screenrc command, or by defining the variable $SCREENCAP prior to startup. When the is latter defined, its value will

be copied verbatim into each window's $TERMCAP variable. This can either be the full terminal definition, or a filename where the terminal ''screen'' (and/or ''screen-w'') is defined.

Note that *screen* honors the ''terminfo'' .screenrc command if the system uses the terminfo database rather than termcap.

When the boolean 'G0' capability is present in the termcap entry for the terminal on which *screen* has been called, the terminal emulation of *screen* supports multiple character sets. This allows an application to make use of, for instance, the VT100 graphics character set or national character sets. The following control functions from ISO 2022 are supported: *lock shift G0* (*SI*), *lock shift G1* (*SO*), *lock shift G2*, *lock shift G3*, *single shift G2*, and *single shift G3*. When a virtual terminal is created or reset, the ASCII character set is designated as *G0* through *G3*. When the 'G0' capability is present, *screen* evaluates the capabilities 'S0', 'E0', and 'C0' if present. 'S0' is the sequence the terminal uses to enable and start the graphics character set rather than *SI*. 'E0' is the corresponding replacement for *SO*. 'C0' gives a character by character translation string that is used during semi-graphics mode. This string is built like the 'acsc' terminfo capability.

When the 'po' and 'pf' capabilities are present in the terminal's termcap entry, applications running in a *screen* window can send output to the printer port of the terminal. This allows a user to have an application in one window sending output to a printer connected to the terminal, while all other windows are still active (the printer port is enabled and disabled again for each chunk of output). As a side-effect, programs running in different windows can send output to the printer simultaneously. Data sent to the printer is not displayed in the window. The *info* command displays a line starting 'PRIN' while the printer is active.

Screen maintains a hardstatus line for every window. If a window gets selected, the display's hardstatus will be updated to match the window's hardstatus line. If the display has no hardstatus the line will be displayed as a standard *screen* message. The hardstatus line can be changed with the ANSI Application Program Command (APC): ''ESC_<string>ESC\''. As a convenience for xterm users the sequence ''ESC]0..2;<string>^G'' is also accepted.

Some capabilities are only put into the $TERMCAP variable of the virtual terminal if they can be efficiently implemented by the physical terminal. For instance, 'dl' (delete line) is only put into the $TERMCAP variable if the terminal supports either delete line itself or scrolling regions. Note that this may provoke confusion, when the session is reattached on a different terminal, as the value of $TERMCAP cannot be modified by parent processes.

The "alternate screen" capability is not enabled by default. Set the **altscreen** .screenrc command to enable it.

The following is a list of control sequences recognized by *screen*. ''(V)'' and ''(A)'' indicate VT100-specific and ANSI- or ISO-specific functions, respectively.

ESC E		Next Line
ESC D		Index
ESC M		Reverse Index
ESC H		Horizontal Tab Set
ESC Z		Send VT100 Identification String
ESC 7	(V)	Save Cursor and Attributes
ESC 8	(V)	Restore Cursor and Attributes
ESC [s	(A)	Save Cursor and Attributes
ESC [u	(A)	Restore Cursor and Attributes
ESC c		Reset to Initial State
ESC g		Visual Bell
ESC Pn **p**		Cursor Visibility (97801)

Pn = **6**		Invisible
7		Visible
ESC =	(V)	Application Keypad Mode
ESC >	(V)	Numeric Keypad Mode
ESC # 8	(V)	Fill Screen with E's
ESC \	(A)	String Terminator
ESC ^	(A)	Privacy Message String (Message Line)
ESC !		Global Message String (Message Line)
ESC k		A. k. a. Definition String
ESC P	(A)	Device Control String. Outputs a string directly to the host terminal without interpretation.
ESC _	(A)	Application Program Command (Hardstatus)
ESC] 0 ; string ^G	(A)	Operating System Command (Hardstatus, xterm title hack)
ESC] 83 ; cmd ^G	(A)	Execute screen command. This only works if multi-user support is compiled into screen. The pseudo-user ''**:window:**'' is used to check the access control list. Use ''addacl :window: -rwx #?'' to create a user with no rights and allow only the needed commands.
Control-N	(A)	Lock Shift G1 (SO)
Control-O	(A)	Lock Shift G0 (SI)
ESC n	(A)	Lock Shift G2
ESC o	(A)	Lock Shift G3
ESC N	(A)	Single Shift G2
ESC O	(A)	Single Shift G3
ESC (Pcs	(A)	Designate character set as G0
ESC) Pcs	(A)	Designate character set as G1
ESC * Pcs	(A)	Designate character set as G2
ESC + Pcs	(A)	Designate character set as G3
ESC [Pn **;** Pn **H**		Direct Cursor Addressing
ESC [Pn **;** Pn **f**		same as above
ESC [Pn **J**		Erase in Display
Pn = None or **0**		From Cursor to End of Screen
1		From Beginning of Screen to Cursor
2		Entire Screen
ESC [Pn **K**		Erase in Line
Pn = None or **0**		From Cursor to End of Line
1		From Beginning of Line to Cursor
2		Entire Line
ESC [Pn **X**		Erase character
ESC [Pn **A**		Cursor Up

ESC [Pn **B**		Cursor Down
ESC [Pn **C**		Cursor Right
ESC [Pn **D**		Cursor Left
ESC [Pn **E**		Cursor next line
ESC [Pn **F**		Cursor previous line
ESC [Pn **G**		Cursor horizontal position
ESC [Pn **‘**		same as above
ESC [Pn **d**		Cursor vertical position
ESC [Ps **;...;** Ps **m**		Select Graphic Rendition
Ps = None or **0**		Default Rendition
1		Bold
2	(A)	Faint
3	(A)	*Standout* Mode (ANSI: Italicized)
4		Underlined
5		Blinking
7		Negative Image
22	(A)	Normal Intensity
23	(A)	*Standout* Mode off (ANSI: Italicized off)
24	(A)	Not Underlined
25	(A)	Not Blinking
27	(A)	Positive Image
30	(A)	Foreground Black
31	(A)	Foreground Red
32	(A)	Foreground Green
33	(A)	Foreground Yellow
34	(A)	Foreground Blue
35	(A)	Foreground Magenta
36	(A)	Foreground Cyan
37	(A)	Foreground White
39	(A)	Foreground Default
40	(A)	Background Black
...		
49	(A)	Background Default
ESC [Pn **g**		Tab Clear
Pn = None or **0**		Clear Tab at Current Position
3		Clear All Tabs
ESC [Pn **;** Pn **r**	(V)	Set Scrolling Region
ESC [Pn **I**	(A)	Horizontal Tab

ESC [Pn **Z**	(A)	Backward Tab
ESC [Pn **L**	(A)	Insert Line
ESC [Pn **M**	(A)	Delete Line
ESC [Pn **@**	(A)	Insert Character
ESC [Pn **P**	(A)	Delete Character
ESC [Pn **S**		Scroll Scrolling Region Up
ESC [Pn **T**		Scroll Scrolling Region Down
ESC [Pn **^**		same as above
ESC [Ps **;...;** Ps **h**		Set Mode
ESC [Ps **;...;** Ps **l**		Reset Mode
Ps = **4**	(A)	Insert Mode
20	(A)	*Automatic Linefeed* Mode
34		Normal Cursor Visibility
?1	(V)	Application Cursor Keys
?3	(V)	Change Terminal Width to 132 columns
?5	(V)	Reverse Video
?6	(V)	*Origin* Mode
?7	(V)	*Wrap* Mode
?9		X10 mouse tracking
?25	(V)	Visible Cursor
?47		Alternate Screen (old xterm code)
?1000	(V)	VT200 mouse tracking
?1047		Alternate Screen (new xterm code)
?1049		Alternate Screen (new xterm code)
ESC [5 i	(A)	Start relay to printer (ANSI Media Copy)
ESC [4 i	(A)	Stop relay to printer (ANSI Media Copy)
ESC [8 ; Ph **;** Pw **t**		Resize the window to ‘Ph’ lines and ‘Pw’ columns (SunView special)
ESC [c		Send VT100 Identification String
ESC [x		Send Terminal Parameter Report
ESC [> c		Send VT220 Secondary Device Attributes String
ESC [6 n		Send Cursor Position Report

INPUT TRANSLATION

In order to do a full VT100 emulation *screen* has to detect that a sequence of characters in the input stream was generated by a keypress on the user’s keyboard and insert the VT100 style escape sequence. *Screen* has a very flexible way of doing this by making it possible to map arbitrary commands on arbitrary sequences of characters. For standard VT100 emulation the command will always insert a string in the input buffer of the window (see also command **stuff** in the command table). Because the sequences generated by a keypress can change after a reattach from a different terminal type, it is possible to bind commands to the termcap name of the keys. *Screen* will insert the correct binding after each reattach. See the **bindkey** command for further details on the syntax and examples.

Here is the table of the default key bindings. (A) means that the command is executed if the keyboard is switched into application mode.

Key name	Termcap name	Command	
Cursor up	ku	stuff \033[A	
		stuff \033OA	(A)
Cursor down	kd	stuff \033[B	
		stuff \033OB	(A)
Cursor right	kr	stuff \033[C	
		stuff \033OC	(A)
Cursor left	kl	stuff \033[D	
		stuff \033OD	(A)
Function key 0	k0	stuff \033[10~	
Function key 1	k1	stuff \033OP	
Function key 2	k2	stuff \033OQ	
Function key 3	k3	stuff \033OR	
Function key 4	k4	stuff \033OS	
Function key 5	k5	stuff \033[15~	
Function key 6	k6	stuff \033[17~	
Function key 7	k7	stuff \033[18~	
Function key 8	k8	stuff \033[19~	
Function key 9	k9	stuff \033[20~	
Function key 10	k;	stuff \033[21~	
Function key 11	F1	stuff \033[23~	
Function key 12	F2	stuff \033[24~	
Home	kh	stuff \033[1~	
End	kH	stuff \033[4~	
Insert	kI	stuff \033[2~	
Delete	kD	stuff \033[3~	
Page up	kP	stuff \033[5~	
Page down	kN	stuff \033[6~	
Keypad 0	f0	stuff 0	
		stuff \033Op	(A)
Keypad 1	f1	stuff 1	
		stuff \033Oq	(A)
Keypad 2	f2	stuff 2	
		stuff \033Or	(A)
Keypad 3	f3	stuff 3	
		stuff \033Os	(A)
Keypad 4	f4	stuff 4	
		stuff \033Ot	(A)
Keypad 5	f5	stuff 5	
		stuff \033Ou	(A)
Keypad 6	f6	stuff 6	
		stuff \033Ov	(A)
Keypad 7	f7	stuff 7	
		stuff \033Ow	(A)
Keypad 8	f8	stuff 8	
		stuff \033Ox	(A)
Keypad 9	f9	stuff 9	
		stuff \033Oy	(A)
Keypad +	f+	stuff +	
		stuff \033Ok	(A)
Keypad -	f-	stuff -	

		stuff \033Om	(A)
Keypad *	f*	stuff *	
		stuff \033Oj	(A)
Keypad /	f/	stuff /	
		stuff \033Oo	(A)
Keypad =	fq	stuff =	
		stuff \033OX	(A)
Keypad .	f.	stuff .	
		stuff \033On	(A)
Keypad ,	f,	stuff ,	
		stuff \033Ol	(A)
Keypad enter	fe	stuff \015	
		stuff \033OM	(A)

SPECIAL TERMINAL CAPABILITIES

The following table describes all terminal capabilities that are recognized by *screen* and are not in the termcap(5) manual. You can place these capabilities in your termcap entries (in '/etc/termcap') or use them with the commands 'termcap', 'terminfo' and 'termcapinfo' in your screenrc files. It is often not possible to place these capabilities in the terminfo database.

LP *(bool)* Terminal has VT100 style margins ('magic margins'). Note that this capability is obsolete because *screen* uses the standard 'xn' instead.

Z0 *(str)* Change width to 132 columns.

Z1 *(str)* Change width to 80 columns.

WS *(str)* Resize display. This capability has the desired width and height as arguments. *SunView(tm)* example: '\E[8;%d;%dt'.

NF *(bool)* Terminal doesn't need flow control. Send ^S and ^Q direct to the application. Same as 'flow off'. The opposite of this capability is 'nx'.

G0 *(bool)* Terminal can deal with ISO 2022 font selection sequences.

S0 *(str)* Switch charset 'G0' to the specified charset. Default is '\E(%.'.

E0 *(str)* Switch charset 'G0' back to standard charset. Default is '\E(B'.

C0 *(str)* Use the string as a conversion table for font '0'. See the 'ac' capability for more details.

CS *(str)* Switch cursor-keys to application mode.

CE *(str)* Switch cursor-keys back to normal mode.

AN *(bool)* Turn on autonuke. See the 'autonuke' command for more details.

OL *(num)* Set the output buffer limit. See the 'obuflimit' command for more details.

KJ *(str)* Set the encoding of the terminal. See the 'encoding' command for valid encodings.

AF *(str)* Change character foreground color in an ANSI conform way. This capability will almost always be set to '\E[3%dm' ('\E[3%p1%dm' on terminfo machines).

AB *(str)* Same as 'AF', but change background color.

AX *(bool)* Does understand ANSI set default fg/bg color (\E[39m / \E[49m).

XC *(str)* Describe a translation of characters to strings depending on the current font. More details follow in the next section.

XT *(bool)* Terminal understands special xterm sequences (OSC, mouse tracking).

C8 *(bool)* Terminal needs bold to display high-intensity colors (e.g. Eterm).

TF *(bool)* Add missing capabilities to the termcap/info entry. (Set by default).

CHARACTER TRANSLATION

Screen has a powerful mechanism to translate characters to arbitrary strings depending on the current font and terminal type. Use this feature if you want to work with a common standard character set (say ISO8851-latin1) even on terminals that scatter the more unusual characters over several national language font pages.

Syntax:

XC=<*charset-mapping*>{**,,**<*charset-mapping*>}
<*charset-mapping*> := <*designator*><*template*>{**,**<*mapping*>}
<*mapping*> := <*char-to-be-mapped*><*template-arg*>

The things in braces may be repeated any number of times.

A <*charset-mapping*> tells *screen* how to map characters in font <*designator*> (’B’: Ascii, ’A’: UK, ’K’: german, etc.) to strings. Every <*mapping*> describes to what string a single character will be translated. A template mechanism is used, as most of the time the codes have a lot in common (for example strings to switch to and from another charset). Each occurrence of ’%’ in <*template*> gets substituted with the <*template-arg*> specified together with the character. If your strings are not similar at all, then use ’%’ as a template and place the full string in <*template-arg*>. A quoting mechanism was added to make it possible to use a real ’%’. The ’\’ character quotes the special characters ’\’, ’%’, and ’,’.

Here is an example:

```
termcap hp700 ’XC=B\E(K%\E(B,\304[,\326\\\\,\334]’
```

This tells *screen* how to translate ISOlatin1 (charset ’B’) upper case umlaut characters on a hp700 terminal that has a german charset. ’\304’ gets translated to ’\E(K[\E(B’ and so on. Note that this line gets parsed *three* times before the internal lookup table is built, therefore a lot of quoting is needed to create a single ’\’.

Another extension was added to allow more emulation: If a mapping translates the unquoted ’%’ char, it will be sent to the terminal whenever *screen* switches to the corresponding <*designator*>. In this special case the template is assumed to be just ’%’ because the charset switch sequence and the character mappings normally haven’t much in common.

This example shows one use of the extension:

```
termcap xterm ’XC=K%,%\E(B,[\304,\\\\\326,]\334’
```

Here, a part of the german (’K’) charset is emulated on an xterm. If *screen* has to change to the ’K’ charset, ’\E(B’ will be sent to the terminal, i.e. the ASCII charset is used instead. The template is just ’%’, so the mapping is straightforward: ’[’ to ’\304’, ’\’ to ’\326’, and ’]’ to ’\334’.

ENVIRONMENT

COLUMNS	Number of columns on the terminal (overrides termcap entry).
HOME	Directory in which to look for .screenrc.
LINES	Number of lines on the terminal (overrides termcap entry).
LOCKPRG	Screen lock program.
NETHACKOPTIONS	Turns on nethack option.
PATH	Used for locating programs to run.

SCREENCAP	For customizing a terminal’s TERMCAP value.
SCREENDIR	Alternate socket directory.
SCREENRC	Alternate user screenrc file.
SHELL	Default shell program for opening windows (default ‘‘/bin/sh’’).
STY	Alternate socket name.
SYSSCREENRC	Alternate system screenrc file.
TERM	Terminal name.
TERMCAP	Terminal description.
WINDOW	Window number of a window (at creation time).

FILES

.../screen-4.?.??/etc/screenrc	
.../screen-4.?.??/etc/etcscreenrc	Examples in the *screen* distribution package for private and global initialization files.
$SYSSCREENRC	
/usr/local/etc/screenrc	*screen* initialization commands
$SCREENRC	
$HOME/.screenrc	Read in after /usr/local/etc/screenrc
$SCREENDIR/S-<login>	
/local/screens/S-<login>	Socket directories (default)
/usr/tmp/screens/S-<login>	Alternate socket directories.
<socket directory>/.termcap	Written by the "termcap" output function
/usr/tmp/screens/screen-exchange	or
/tmp/screen-exchange	*screen* ‘interprocess communication buffer’
hardcopy.[0-9]	Screen images created by the hardcopy function
screenlog.[0-9]	Output log files created by the log function
/usr/lib/terminfo/?/*	or
/etc/termcap	Terminal capability databases
/etc/utmp	Login records
$LOCKPRG	Program that locks a terminal.

SEE ALSO

termcap(5), utmp(5), vi(1), captoinfo(1), tic(1)

AUTHORS

Originally created by Oliver Laumann, this latest version was produced by Wayne Davison, Juergen Weigert and Michael Schroeder.

COPYLEFT

02111-1307, USA

CONTRIBUTORS

Ken Beal (kbeal@amber.ssd.csd.harris.com),
Rudolf Koenig (rfkoenig@immd4.informatik.uni-erlangen.de),
Toerless Eckert (eckert@immd4.informatik.uni-erlangen.de),
Wayne Davison (davison@borland.com),
Patrick Wolfe (pat@kai.com, kailand!pat),
Bart Schaefer (schaefer@cse.ogi.edu),
Nathan Glasser (nathan@brokaw.lcs.mit.edu),
Larry W. Virden (lvirden@cas.org),
Howard Chu (hyc@hanauma.jpl.nasa.gov),
Tim MacKenzie (tym@dibbler.cs.monash.edu.au),
Markku Jarvinen (mta@{cc,cs,ee}.tut.fi),
Marc Boucher (marc@CAM.ORG),
Doug Siebert (dsiebert@isca.uiowa.edu),
Ken Stillson (stillson@tsfsrv.mitre.org),
Ian Frechett (frechett@spot.Colorado.EDU),
Brian Koehmstedt (bpk@gnu.ai.mit.edu),
Don Smith (djs6015@ultb.isc.rit.edu),
Frank van der Linden (vdlinden@fwi.uva.nl),
Martin Schweikert (schweik@cpp.ob.open.de),
David Vrona (dave@sashimi.lcu.com),
E. Tye McQueen (tye%spillman.UUCP@uunet.uu.net),
Matthew Green (mrg@eterna.com.au),
Christopher Williams (cgw@pobox.com),
Matt Mosley (mattm@access.digex.net),
Gregory Neil Shapiro (gshapiro@wpi.WPI.EDU),
Johannes Zellner (johannes@zellner.org),
Pablo Averbuj (pablo@averbuj.com).

VERSION

This is version 4.0.2. Its roots are a merge of a custom version 2.3PR7 by Wayne Davison and several enhancements to Oliver Laumann's version 2.0. Note that all versions numbered 2.x are copyright by Oliver Laumann.

AVAILABILITY

The latest official release of *screen* available via anonymous ftp from gnudist.gnu.org, nic.funet.fi or any other *GNU* distribution site. The home site of *screen* is ftp.uni-erlangen.de, in the directory pub/utilities/screen. The subdirectory 'private' contains the latest beta testing release. If you want to help, send a note to screen@uni-erlangen.de.

BUGS

- 'dm' (delete mode) and 'xs' are not handled correctly (they are ignored). 'xn' is treated as a magic-margin indicator.
- *Screen* has no clue about double-high or double-wide characters. But this is the only area where *vttest* is allowed to fail.
- It is not possible to change the environment variable $TERMCAP when reattaching under a different terminal type.
- The support of terminfo based systems is very limited. Adding extra capabilities to $TERMCAP may not have any effects.

- *Screen* does not make use of hardware tabs.
- *Screen* must be installed as set-uid with owner root on most systems in order to be able to correctly change the owner of the tty device file for each window. Special permission may also be required to write the file ‘‘/etc/utmp’’.
- Entries in ‘‘/etc/utmp’’ are not removed when *screen* is killed with SIGKILL. This will cause some programs (like "w" or "rwho") to advertise that a user is logged on who really isn’t.
- *Screen* may give a strange warning when your tty has no utmp entry.
- When the modem line was hung up, *screen* may not automatically detach (or quit) unless the device driver is configured to send a HANGUP signal. To detach a *screen* session use the -D or -d command line option.
- If a password is set, the command line options -d and -D still detach a session without asking.
- Both ‘‘breaktype’’ and ‘‘defbreaktype’’ change the break generating method used by all terminal devices. The first should change a window specific setting, where the latter should change only the default for new windows.
- When attaching to a multiuser session, the user’s .screenrc file is not sourced. Each user’s personal settings have to be included in the .screenrc file from which the session is booted, or have to be changed manually.
- A weird imagination is most useful to gain full advantage of all the features.
- Send bug-reports, fixes, enhancements, t-shirts, money, beer & pizza to **screen@uni-erlangen.de**.

NAME

tcpprof − report profile of network traffic

SYNOPSIS

tcpprof [**-?hdnpR**] [**-f** *filter expr*] [**-i** *interface*] [**-P** *port*] [**-r** *filename*] [**-s** *seconds*] [**-S** *letters*] [**-t** *lines*]

DESCRIPTION

tcpprof reports a profile of network traffic by ranking it by link type, ip protocol, TCP/UDP port, ip address, or network address.

Network information is collected either by reading data from *filename*, or by directly monitoring the network interface *interface*. The default action for **tcpprof** is to automatically search for an appropriate interface, and to generate a profile before it exits.

When reading data from *filename*, **tcpprof** will display the profile and exit immediately after the entire file has been processed. When collecting data from *interface*, **tcpprof** will keep running unless the **-s** option had been specified.

OPTIONS

The options are as follows:

-f *filter expr*

Filter the packets according the rules given by *filter expr*. For the syntax of these rules, see tcpdump(1). The argument must be quoted if it contains spaces in order to separate it from other options.

-h, **-**? Display version and a brief help message.

-d **tcpprof** will track the source and destination information separately, where applicable, and identify source data with a ">" and destination data with "<". For example, a "http <" statistic signifies all traffic with destination port 80 (http). This option only applies to port, host and network statistics.

-i *interface*

Do a live capture (rather than read from a file) on the interface *interface* given on the command line. If *interface* is "auto" then **tcpprof** tries to find an appropriate one by itself.

-P *port* This tells **tcpprof** to ignore TCP and UDP ports greater than or equal to *port* when displaying port statistics. This is not the same as filtering these port numbers out of the data set. This way, packets with i.e. the source port above *port* and the destination port below *port* will be able to still count the lower port number as a statistic. In addition, this doesn't affect the other statistic types (link, protocol, etc.)

-p Set the interface into non-promiscuous mode (promiscuous is the default) when doing live captures.

-r *filename*

Read all data from *filename*, which may be a regular file, a named pipe or "-" to read it's data from standard input. Acceptable file formats include pcap (tcpdump(1) files) and "snoop" format files. *filename* is usually a file created by the tcpdump(1) command using the "-w" option.

-S *letters*

Tells **tcpprof** which statistics to display. *letters* must be a string of one or more of the following letters:

l show stats about the link layer

i show stats about all ip protocols

p show stats about TCP/UDP ports

h show stats about hosts/ip addresses

n show stats about network addresses

a a synonym for "liphn"

-s *seconds*
When monitoring an interface, **tcpprof** runs for only *seconds* seconds, and then quits. Has no effect when reading data from a file.

-t *lines* When printing a profile of the data, **tcpprof** will display a maximum of *lines* lines for each statistic.

SIGNALS

Upon receiving a SIGINT, **tcpprof** will print any remaining statistics, and then exit.

FILES

`/dev/bpf`**n** the packet filter device

EXAMPLES

```
tcpprof -i fxp0 -S a
```

Displays a complete profile of network data passing through the fxp0 network interface, after the user enters ^C (control C).

```
tcpprof -r file.dump -S a
```

Displays a complete profile of network data from the `tcpdump`(1) generated file "file.dump".

SEE ALSO

`tcpdump`(1), `pcap`(3), `bpf`(4)

HISTORY

tcpprof was first written along side tcpstat in Winter 1998 using FreeBSD 3.0, and then finaly retrofited for Linux in Spring 2000. It became installed along with tcpstat since version 1.5.

AUTHORS

Paul Herman ⟨pherman@frenchfries.net⟩
Cologne, Germany.

Please send all bug reports to this address.

BUGS

Not tested with link types other than Ethernet, PPP, and "None" types.

There may be problems reading non-IPv4 packets across platforms when reading null type link layers. This is due to a lack of a standardized packet type descriptor in libpcap for this link type.

Snoop file formats cannot be read from stdin or named pipes.

NAME

tcpstat − report network interface statistics

SYNOPSIS

tcpstat [**−?haeFlp**] [**−B** *bps*] [**−b** *bps*] [**−f** *filter expr*] [**−i** *interface*]
[**−o** *output*] [**−R** *seconds*] [**−r** *filename*] [**−s** *seconds*] [*interval*]

DESCRIPTION

tcpstat reports certain network interface statistics much like vmstat(8) does for system statistics. Statistics include bandwidth being used, number of packets, average packet size, and much more.

Network information is collected either by reading data from *filename*, or by directly monitoring the network interface *interface*. The default action for **tcpstat** is to automatically search for an appropriate interface, and to show current statistics on it.

interval is the sample interval, in seconds, in which the statistics are based upon and when in default mode, how often the display is updated. If -1 is given, then the interval is taken to be the entire length of the sample. Default is 5 seconds.

When reading data from *filename*, **tcpstat** will exit immediately after the entire file has been processed. When collecting data from *interface*, **tcpstat** will keep running unless the **−s** option had been specified.

OPTIONS

The options are as follows:

−a Accounting mode. Displays the estimated number of bytes per second, minute, hour, day, and month.

−b *bps* Bandwidth mode. Displays the total number of seconds the data-throughput exceeded *bps*, and the percentage of total time this was, as if the interface were limited to *bps* bits per second. See the **NOTES** section below to see how the *interval* affects bandwidth calculation.

−B *bps* "Dumb" bandwidth mode. Displays the total number of seconds the data-throughput exceeded *bps*, and the percentage of total time this was. See the **NOTES** section below to see difference between "dumb" and normal bandwidth modes.

−e Suppresses the display of empty intervals.

−F Flush all output streams after printing each interval. Sometimes useful when redirecting output into a file, or piping tcpstat into another program like grep(1).

−f *filter expr*
Filter the packets according the rules given by *filter expr*. For the syntax of these rules, see tcpdump(1). The argument must be quoted if it contains spaces in order to separate it from other options.

−h, **−**? Display version and a brief help message.

−i *interface*
Do a live capture (rather than read from a file) on the interface *interface* given on the command line. If *interface* is "auto" then **tcpstat** tries to find an appropriate one by itself.

−l Include the size of the link-layer header when calculating statistics. (Ethernet only, right now. Usually 14 bytes per packet.)

-p Set the interface into non-promiscuous mode (promiscuous is the default) when doing live captures.

-o *format*
Set the output format when displaying statistics. See the **OUTPUT FORMAT** section below for a description of the syntax.

-R *seconds*
Show the timestamp relative to *seconds*. Avoid this option, because it will most likely go away in future versions.

-r *filename*
Read all data from *filename*, which may be a regular file, a named pipe or "-" to read it's data from standard input. Acceptable file formats include pcap (`tcpdump`(1) files) and "snoop" format files. *filename* is usually a file created by the `tcpdump`(1) command using the "-w" option.

-s *seconds*
When monitoring an interface, **tcpstat** runs for only *seconds* seconds, and then quits. When reading from a data file, **tcpstat** prints statistics for *seconds* seconds relative to the first packet seen.

OUTPUT FORMAT

The *output* string is any quoted string, and **tcpstat** will write this string to the stdout. In addition, **tcpstat** will substitute certain values for substrings which begin with a "%", as well as most standard `printf`(3) "\" escape characters. Here is a list of all substitution strings:

%A the number of ARP packets

%a the average packet size in bytes

%B the number of bytes per second

%b the number of bits per second

%C the number of ICMP and ICMPv6 packets

%d the standard deviation of the size of each packet in bytes

%I the number of IPv4 packets

%l the network "load" over the last minute, similar to `uptime`(1)

%M the maximum packet size in bytes

%m the minimum packet size in bytes

%N the number of bytes

%n the number of packets

%p the number of packets per second

%R same as %S, but relative to the first packet seen

%r same as %s, but relative to the first packet seen

%S the timestamp for the interval in seconds after the "UNIX epoch"

%s the timestamp for the interval in seconds.microseconds after the "UNIX epoch"

%T the number of TCP packets

%U the number of UDP packets

%V the number of IPv6 packets

%*number*

switch the output to the file descriptor *number* at this point in the string. All output for each interval before this parameter is by default the standard output (file descriptor 1). Useful when redirecting the output into more than one file (or fifo) for separate statistics. Be sure you know where they are going. Writing to "dangling" file descriptors (without directing them to a specific destination) may produce unexpected results.

%% the "%" character

The default *format* string for **tcpstat** is:

"Time:%S\tn=%n\tavg=%a\tstddev=%d\tbps=%b\n"

which will produce an output which would look similar to:

```
Time:940948785 n=107   avg=251.81     stddev=422.45  bps=43110.40
Time:940948790 n=99    avg=400.21     stddev=539.39  bps=63393.60
Time:940948795 n=43    avg=257.16     stddev=352.83  bps=17692.80
```

It is worth noting for example, that many of the protocol filters (%T, %U, etc.) may be seen as being redundant because protocols can be filtered using **-f** (see **OPTIONS** above)

SIGNALS

Upon receiving a SIGINT, **tcpstat** will print any remaining statistics, and then exit. Upon receiving a SIGUSR1 when printing intervals, **tcpstat** will print the current statistics immediately. This can be useful when using an interval length of "-1" to print statistics on demand.

FILES

`/dev/bpf`**n** the packet filter device

EXAMPLES

```
tcpstat -i fxp0
```

Displays the default statistics every 5 seconds of all traffic currently passing through the fxp0 network interface.

```
tcpstat -r file.dump
```

Displays the default statistics every 5 seconds from the `tcpdump`(1) generated file "file.dump".

```
tcpstat -f 'port (smtp or http)' -o '%S %b\n' -r file.dump 2.3
```

Displays every 2.3 seconds the timestamp together with smtp and http traffic throughput of the data from "file.dump", in a format which would be suitable for `gnuplot`(1).

```
tcpstat -b 28800 -r file.dump 0.5
```

Displays what percentage of the traffic in file.dump exceeded the speed of my modem (28800 bits per second.)

SEE ALSO

`tcpdump`(1), `pcap`(3), `bpf`(4), `printf`(3)

NOTES

Interval size affects bandwidth

Due to the nature of how bandwidth is actually measured (from discrete samples of data), the bandwidth numbers displayed will vary according to the *interval* variable. Generally speaking, if you often have rapid bursts of packet data, the bandwidth reported will not reflect this when *interval* is sufficiently large. This results in an "averaging" effect, which may or may not be desired. On the other hand, if *interval* is too small (say < 0.01), this results in unrealisticaly large bandwidths for very short amounts of time.

The reason for the latter is that most network interfaces do not hand over packets bit by bit, but rather packet by packet. Thus, each packet is reported as being tranfered "instantaneously", resulting in "infinite" (or rather indeterminable) bandwidth. Thus, when counting single bits on the wire, there really is no such thing as "bandwidth" because they aren't really moving from the network stack's point of view (cf. Zeno's Paradox.)

A possible solution is to internaly spline the packet sizes together and report the bandwidth as the scalar integral over the given interval, but this has yet to be implimented, and to be honest, would be the proverbial cruise missle to destroy an ant hill.

That being said (whew!), a "good value" for *interval* is usualy somewhere between 0.5 and 2.

Difference between normal and 'dumb' bandwidth modes.

In normal bandwidth mode, when an interval exceeds the given bandwidth, the extra bytes are "moved" into the next interval. This has the effect of trying to imagine how overloaded an interface would be if the interface had a smaller bandwidth, yet same amount of data tried to get through.

In "dumb" bandwidth mode, each interval which exceeds the given bandwidth is simply counted. Nothin' else.

HISTORY

tcpstat was first written in Winter 1998 using FreeBSD 3.0, and then finaly retrofited for Linux in Spring 2000.

AUTHORS

Paul Herman ⟨pherman@frenchfries.net⟩
Cologne, Germany.

Please send all bug reports to this address.

BUGS

Due to a bug in libpcap, tcpstat will hang indefinately under Linux when no packets arrive. This is because the timeout in pcap_open_live() is ignored under Linux when the interface is idle, which causes pcap_dispatch() to never return.

Not tested with link types other than Ethernet, PPP, and "None" types.

There may be problems reading non-IPv4 packets across platforms when reading null type link layers. This is due to a lack of a standardized packet type descriptor in libpcap for this link type.

Snoop file formats cannot be read from stdin or named pipes.

NAME

cbm − display the current traffic on all network devices

SYNOPSIS

cbm [**−−help**] [**−−version**]

DESCRIPTION

cbm the Color Bandwidth Meter displays the current traffic on all network devices.

This program is so simple that it should be self−explanatory.

OPTIONS

−−help Display some help and exit.

−−version
Display version information and exit.

INTERACTIVE CONTROL

cbm can be controlled with the following keys:

Up/Down
Select an interface to show details about.

q Exit from the program.

b Switch between bits per second and bytes per second.

\+ Increase the update delay by 100ms.

− Decrease the update delay by 100ms.

AUTHOR

Source code and newer versions are available from *http://www.isotton.com/utils/cbm/*.

Report bugs to <aaron@isotton.com> with ‘‘[cbm]’’ as the first word of the subject.

NAME

realpath − Return the real path of a link

SYNOPSIS

realpath *path path path ...*

PACKAGE

UNItools

DESCRIPTION

realpath program returns the real path to something pointed to, by a symbolic link. E.g.:

```
ln -s /etc/passwd /tmp/passwd
ln -s /tmp/passwd /tmp/passwd-link-level2
realpath /tmp/passwd-link-level2
```

/etc/passwd

OPTIONS

path One or more pathname(s).

DIAGNOSTICS

Missing file, no permissions, etc. goes to STDOUT.

ENVIRONMENT

SEE ALSO

`realpath(3c)`

VERSION

$Date: 2003/10/24 12:59:21 $
$Revision: 1.1 $
$Source: /lan/ssi/projects/UNItools/Linux/src/realpath/RCS/realpath.c,v $
$State: Exp $

HISTORY

See `rlog realpath.c`.

BUGS

Probably. Please report them to the call-desk or the author.

AUTHOR(S)

Niels Thomas Haug?
E-mail: thomas@haugaard.net
UNI•C
DTU, Building 304
DK-2800 Kgs. Lyngby
Denmark

NAME

iftop - display bandwidth usage on an interface by host

SYNOPSIS

iftop -h | [**-nNpbBP**] [**-i** *interface*] [**-f** *filter code*] [**-F** *net*/*mask*]

DESCRIPTION

iftop listens to network traffic on a named *interface*, or on the first interface it can find which looks like an external interface if none is specified, and displays a table of current bandwidth usage by pairs of hosts. **iftop** must be run with sufficient permissions to monitor all network traffic on the *interface*; see **pcap**(3) for more information, but on most systems this means that it must be run as root.

By default, **iftop** will look up the hostnames associated with addresses it finds in packets. This can cause substantial traffic of itself, and may result in a confusing display. You may wish to suppress display of DNS traffic by using filter code such as **not port domain**, or switch it off entirely, by using the **-n** option or by pressing **R** when the program is running.

By default, **iftop** counts all IP packets that pass through the filter, and the direction of the packet is determined according to the direction the packet is moving across the interface. Using the **-N** option it is possible to get **iftop** to show packets entering and leaving a given network. For example, **iftop -N 10.0.0.0/255.0.0.0** will analyse packets flowing in and out of the 10.* network.

Some other filter ideas:

not ether host ff:ff:ff:ff:ff:ff
: Ignore ethernet broadcast packets.

port http and not host *webcache.example.com*
: Count web traffic only, unless it is being directed through a local web cache.

icmp How much bandwith are users wasting trying to figure out why the network is slow?

OPTIONS

-h Print a summary of usage.

-n Don't do hostname lookups.

-N Do not resolve port number to service names

-p Run in promiscuous mode, so that traffic which does not pass directly through the specified interface is also counted.

-P Turn on port display.

-b Don't display bar graphs of traffic.

-B Display bandwidth rates in bytes/sec rather than bits/sec.

-i *interface*
: Listen to packets on *interface*.

-f *filter code*
: Use *filter code* to select the packets to count. Only IP packets are ever counted, so the specified code is evaluated as **(***filter code***) and ip**.

-F *net*/*mask*
: Specifies a network for traffic analysis. If specified, iftop will only include packets flowing in to or out of the given network, and packet direction is determined relative to the network boundary, rather than to the interface. You may specify *mask* as a dotted quad, such as /255.255.255.0, or as a single number specifying the number of bits set in the netmask, such as /24.

-c *config file*
Specifies an alternate config file. If not specified, iftop will use **~/.iftopc** if it exists. See below for a description of config files

DISPLAY

When running, **iftop** uses the whole screen to display network usage. At the top of the display is a logarithmic scale for the bar graph which gives a visual indication of traffic.

The main part of the display lists, for each pair of hosts, the rate at which data has been sent and received over the preceding 2, 10 and 40 second intervals. The direction of data flow is indicated by arrows, <= and =>. For instance,

```
foo.example.com  =>  bar.example.com     1Kb  500b   100b
                 <=                      2Mb    2Mb    2Mb
```

shows, on the first line, traffic from **foo.example.com** to **bar.example.com**; in the preceding 2 seconds, this averaged 1Kbit/s, around half that amount over the preceding 10s, and a fifth of that over the whole of the last 40s. During each of those intervals, the data sent in the other direction was about 2Mbit/s. On the actual display, part of each line is inverted to give a visual indication of the 10s average of traffic. You might expect to see something like this where host **foo** is making repeated HTTP requests to **bar**, which is sending data back which saturates a 2Mbit/s link.

By default, the pairs of hosts responsible for the most traffic (10 second average) are displayed at the top of the list.

At the bottom of the display, various totals are shown, including peak traffic over the last 40s, total traffic transferred (after filtering), and total transfer rates averaged over 2s, 10s and 40s.

SOURCE / DEST AGGREGATION

By pressing **s** or **d** while **iftop** is running, all traffic for each source or destination will be aggregated together. This is most useful when **iftop** is run in promiscuous mode, or is run on a gateway machine.

PORT DISPLAY

S or **D** toggle the display of source and destination ports respectively. **p** will toggle port display on/off.

DISPLAY TYPE

t cycles through the four line display modes; the default 2-line display, with sent and received traffic on separate lines, and 3 1-line displays, with sent, received, or total traffic shown.

DISPLAY ORDER

By default, the display is ordered according to the 10s average (2nd column). By pressing **1**, **2** or **3** it is possible to sort by the 1st, 2nd or 3rd column. By pressing < or > the display will be sorted by source or destination hostname respectively.

DISPLAY FILTERING

l allows you to enter a POSIX extended regular expression that will be used to filter hostnames shown in the display. This is a good way to quickly limit what is shown on the display. Note that this happens at a much later stage than filter code, and does not affect what is actually captured. Display filters DO NOT affect the totals at the bottom of the screen.

PAUSE DISPLAY / FREEZE ORDER

P will pause the current display.

o will freeze the current screen order. This has the side effect that traffic between hosts not shown on the screen at the time will not be shown at all, although it will be included in the totals at the bottom of the screen.

SCROLL DISPLAY

j and **k** will scroll the display of hosts. This feature is most useful when the display order is frozen (see above).

FILTER CODE

f allows you to edit the filter code whilst iftop running. This can lead to some unexpected behaviour.

CONFIG FILE

iftop can read its configuration from a config file. If the **-c** option is not specified, iftop will attempt to read its configuration from **~/.iftoprc**, if it exists. Any command line options specified will override settings in the config file.

The config file consists of one configuration directive per line. Each directive is a name value pair, for example:

interface: eth0

sets the network interface. The following config directives are supported:

interface: *if*
Sets the network interface to *if*.

dns-resolution: *(yes|no)*
Controls reverse lookup of IP addresses.

port-resolution: *(yes|no)*
Controls conversion of port numbers to service names.

filter-code: *bpf*
Sets the filter code to *bpf*.

show-bars: *(yes|no)*
Controls display of bar graphs.

promiscuous: *(yes|no)*
Puts the interface into promiscuous mode.

port-display: *(off|source-only|destination-only|on)*
Controls display of port numbers.

hide-source: *(yes|no)*
Hides source host names.

hide-destination: *(yes|no)*
Hides destination host names.

use-bytes: *(yes|no)*
Use bytes for bandwidth display, rather than bits.

sort: *(2s|10s|40s|source|destination)*
Sets which column is used to sort the display.

line-display: *(two-line|one-line-both|one-line-sent|one-line-received)*
Controls the appearance of each item in the display.

show-totals: *(yes|no)*
Shows cummulative total for each item.

log-scale: *(yes|no)*
Use a logarithmic scale for bar graphs.

max-bandwidth: *bw*
Fixes the maximum for the bar graph scale to *bw*, e.g. "10M"

net-filter: *net/mask*
Defines an IP network boundary for determining packet direction.

screen-filter: *regexp*
Sets a regular expression to filter screen output.

QUIRKS (aka they're features, not bugs)

There are some circumstances in which iftop may not do what you expect. In most cases what it is doing is logical, and we believe it is correct behaviour, although I'm happy to hear reasoned arguments for alternative behaviour.

Totals don't add up

There are several reasons why the totals may not appear to add up. The most obvious is having a screen filter in effect, or screen ordering frozen. In this case some captured information is not being shown to you, but is included in the totals.

A more subtle explanation comes about when running in promiscuous mode without specifying a **-N** option. In this case there is no easy way to assign the direction of traffic between two third parties. For the purposes of the main display this is done in an arbitrary fashion (by ordering of IP addresses), but for the sake of totals all traffic between other hosts is accounted as incoming, because that's what it is from the point of view of your interface. The **-N** option allows you to specify an arbitrary network boundary, and to show traffic flowing across it.

Peak totals don't add up

Again, this is a feature. The peak sent and peak received didn't necessarily happen at the same time. The peak total is the maximum of sent plus received in each captured time division.

Changing the filter code doesn't seem to work

Give it time. Changing the filter code affects what is captured from the time that you entered it, but most of what is on the display is based on some fraction of the last 40s window of capturing. After changing the filter there may be entries on the display that are disallowed by the current filter for up to 40s. DISPLAY FILTERING has immediate effect and does not affect what is captured.

FILES

~/.iftoprc
Configuration file for iftop.

SEE ALSO

tcpdump(8), **pcap**(3), **driftnet**(1).

AUTHOR

Paul Warren <pdw@ex-parrot.com>

VERSION

$Id: iftop.8,v 1.24 2003/10/22 19:28:31 pdw Exp $

COPYING

Object Filler & Object Dumper
Version 2.0

User's Manual

Martín Humberto Hoz Salvador
mhoz@mexico.com

February 12th, 2005
Revision B

TABLE OF CONTENTS

Disclaimer, License of use and Limit of Liability

The programs Object Filler, Object Dumper and its companions described on this documentation are not official nor supported in any form, expressed or implicit, by any entity.

The above statement includes that Check Point Software Technologies does not support nor backs these tools. These tools were made by a merely personal initiative as a technology proof-of-concept, neither for profit nor for financial gain in any form.

These programs come with no support of any kind and with no warranty of any kind. You (the user of it) are responsible for any use these programs may have, good or bad; and for any other good or bad thing derived of this program anyhow while you use it directly or indirectly (including but not limited to data loss, programs failure of any kind or promotions). The author(s), their employers and any other entity with direct or indirect relationship to them, are not liable in any way or form because of the misuse, abuse or use of the mentioned programs. The author(s) have made no warranty nor representation that the operation of this software will be error-free or suitable for any direct or indirect application, and they are under no obligation to provide any services, by way of maintenance, update, or otherwise. This software is an always experimental prototype offered on an as-is basis; and should be treated accordingly.

You are not allowed to disassemble, reverse engineer or perform any other known or unknown way to discover the mechanisms used by the tools, over the binary forms. You may also not use these tools, the knowledge of them, to harm in any known or unknown way to the author, Check Point Software Technologies, or any other entity with relationship to them.

You may use the programs Object Filler, Object Dumper and its companions described on this documentation, free of charge (free of cost), but if you are not a Check Point Software Technologies employee, you cannot redistribute them in any form. Check Point Software Technologies employees are allowed to redistribute this software, as long the original program package and documentation is preserved. If you want to share these programs, you must direct the interested entity to download it from any of the sites where they are available.

All other brands and commercial trademarks mentioned on this documentation, and in the programs described by this documentation, are property of their respective owners.

Using the tools in any way, makes you implicitly accept the terms of use above listed.

Introduction

Administrators using Check Point products and maintaining the SmartCenter server, may find the graphical interface provided to manage their security configuration very appealing for the day-to-day operations. However, from time to time there are needs where a graphical interface may fall a bit short, or where you may wish additional functionality today not freely available. Samples of these cases may be:

1.- You are tasked by a customer to configure a Check Point solution where creating all the networks from 10.100.0.0 to 10.100.255.0 with a 24 bit netmask (255.255.255.0) is needed. This means you'll have to manually click to create 256 objects. If all of them will be NATted using Hide NAT and the IP address 1.2.3.4, them more clicks are needed when generating the objects. This may be painful and time consuming

2.- You're migrating from a Cisco PIX or Cisco Router ACL, Juniper/NetScreen, Symantec's Raptor, SecureComputing's SideWinder or Gauntlet to a Check Point solution. You have tons of rules already created and you want to save some time while passing this information from the old platform to your brand new Check Point.

3.- You or your customer has 50 IP addresses (or more) assigned to internal users or services, on a Microsoft Excel sheet. Those IP addresses are not on any given order, so you cannot create networks or ranges, and it's needed to create individual host objects representing such workstations for granular access control policies. This may take some time and can be error-prone if done via a Graphical Interface.

For all those cases, it's possible to save some time using a marvelous command line interface that is installed with all the Check Point SmartCenters on any supported platform. This command line interface is supported by Check Point and is called dbedit.

dbedit is then, a very powerful command-line based tool that allows to control the Check Point's SmartCenter object database (which you can find in a somehow human-readable text file on $FWDIR/conf/objects_5_0.C on SUN Solaris, Nokia IPSO, Red Hat Linux, Check Point SecurePlatform, or %FWDIR%\conf\objects_5_0.C in Microsoft Windows)

Object Filler and Object Dumper are tools oriented to Security Administrators or Security Engineers using Check Point Products, which ease the use of dbedit to perform administrative tasks

Object Filler is an automated tool that can take a couple of IP addresses and a netmask as entry from the command line, or information from a file with certain format, and then produces dbedit commands that automatically generate network objects, services and rules, easing the task of populating the SmartCenter with the information you need.

In the other hand, Object Dumper does somehow the opposite: given an Objects_5_0.C file, Object Dumper can export the content to a CSV file which you can review or modify using any spreadsheet program able to understand CSV (Comma Separated Values). Microsoft Excel is an example. If you want, you can modify such CSV file, and then import back your modifications to the SmartCenter Server using Object Filler.

Object Dumper also gives you the chance of exporting the Objects_5_0.C file to an HTML table for documentation purposes, even though I would recommend using some other program to do that (please refer to the FAQ section of this document for a list of suggested programs).

This manual describes how these tools work. Please read this documentation before sending a question to the author or any other entity, such as mailing lists.

Contacting the Author

If you wish to contact the author of the tools, you may send an e-mail to mhoz@mexico.com with the subject *About Object Filler*. I always read and answer my e-mails, and I always take in account the suggestions given.

If for any reason, the communication requires to transfer some sensitive files or messages, you may encrypt such content using my PGP public key included in the distribution files. If for any reason you have no access to such file, you may locate it in a public key server, under the PGP Key ID 0x0454E8D9

I strongly encourage you to send me an e-mail if you are successfully using these tools in your environment. So far, the amount of feedback I get is very low, and would like to know more of any use you may have given to the tools, any stories (good or bad) around them, suggestions of course, and overall what do you think on them in general.

Related programs

There are several programs that do tasks similar or related to what Object Filler and Object Dumper do. Some of them are supported by Check Point. The following is an incomplete list of such programs, that you may find useful in tasks Object Filler/Object Dumper are not designed to do:

- **Web Visualization Tool:**
  http://www.checkpoint.com/techsupport/downloadsng/utilities.html#visualization
  Officially supported by Check Point, that supports exporting rules and objects to HTML/XML format.
- **FW1Rules:**
  http://www.wyae.de/software/fw1rules/
  Unsupported by Check Point tool, that allows to export both objects and rules in several formats, including HTML and CSV. Written on PERL.
- **CPrules:**
  http://www.wormnet.nl/cprules/
  Unsupported by Check Point tool that allows to export the rulebase and objects to HTML. Tool especially enhanced for NG installations. Written on PERL.
- **Upgrade tools:**
  http://www.checkpoint.com/techsupport/downloadsng/utilities.html#upgrade_verify
  Supported by Check Point tools that help to migrate from previous versions to the current one. These tools also proactively tell of potential problems (and the solution) when upgrading. Besides that, they can export and import the whole configuration in different machines and even across platforms. Documentation included in the package.
- **CP Merge:**
  http://www.checkpoint.com/techsupport/downloadsng/utilities.html#cpmerge
  Supported by Check Point tools that help import, export and merge policies from and to the SmartCenter. Documentation included in the package.
- **fw dbimport/fw dbexport:**
  Supported by Check Point command line tools included in the product, that help to import and export user's information. The documentation for such commands is available at the Docs directory in the CD where the Check Point's software comes, and the filename is CommandLineInterface.pdf. You may find the Online version of the manual here (Valid Software Subcription Account is required):
  http://www.checkpoint.com/support/downloads/docs/firewall1/r55/CommandLineInterface.PDF
  http://www.checkpoint.com/support/downloads/docs/firewall1/r54/CommandLineInterface.pdf

Object Filler & Object Dumper Programs Availability

Object Filler and Object Dumper have no associated cost (i.e. no license fee) – as they are not part of the official Check Point Software.

Latest versions are always available at the following sites (they are not listed on any particular order):
http://www.phoneboy.com - under *FireWall-1 FAQs* and then *Software Downloads*
http://lindos.dnsalias.com/
http://www.lindercentral.com/ofiller/
http://mhoz1.sofaware.net/ofiller/ - Unreliable Server

These programs are also available through Check Point System Engineers world wide on a non-official way - i.e. they are not obligated to provide it to you, they are also not obligated to support you if you run into problems. They are not even obligated to know that these tools exist.

I would like to thank very, very, very (very) much to Jeff Mousseau, Nokia SE at Canada, for letting me use his website (http://www.digitalmigrations.com) as the official download site for the tools for two years (2003-2004). Such a reliable and well organized website for sure will be missed by the security community worldwide.

I also want give my sincere and deep thanks to Brian Linder, Pedro Paixão and Dameon D. Welch-Abernathy (a.k.a. phoneboy) for hosting the latest version of the tools in their websites.

Acknowledgements

I would like to thank the following individuals for their direct and indirect help with this: providing information to me, testing it, telling me it actually works! (or that actually have bugs) ;-), commenting and suggesting features or utilization scenarios, and overall helping me on improving both programs – I'm sure I'm still missing people, so I apologize in advance for that.

Adrián Espinosa, Andrew Singer, Amir Kossover, Arturo Gómez, Brian Linder, Chris Lyttle, Chris Tobkin, Dameon D. Welch-Abernathy, Damien DeVille, David Hernández, Diego Lastra, Dino Constantinou, Elad Lavi, Eli Faskha, Emilio Sánchez, Enaela García, Erez Shtang, Fernando Acosta, Gil Sudai, Gil Shapira, Héctor Garza, Idan Plotnik, Jaime Castañeda, Jeff Mousseau, Jim Hebert, José Agüero, Juan Garza, Julio Salas, Kellman Meghu, Kris Boulez, Leonid Belkind, Marc Gorelick, Marc Lampo, Mario Cinco, Mario Garibay, Mats Ekdahl, Ofer Barzvi, Ofir Barzilai, Oscar Viniegra, Paul Frumer, Pedro Paixão, Peter Sandkuijl, René Tavera, Rickardo González, Rob Sparre, Rodrigo Díaz, Ronen Leshem, Sharon Besser, Tal Shevach, Tom Calarco, Udo Schneider, Zohar Erel.

To all of you, my deepest and sincere thanks for helping me to bring the tools to the current stage in one way or another.

Program History

* Version 2.0 – February 2005

- Object Filler
  - Added support for ICMP, Other, RPC and DCE-RPC services in CSV files.
  - Added support for Check Point Dynamic Gateways (Check Point Gateways with Dynamic IP address) in CSV Files
  - Enhanced support for Raptor files, including services recognition
  - Added support for interfaces on Check Point gateways (including Clusters and Dynamic IP gateways) and interoperable devices. Interfaces with dynamic IP are also supported. This is in CSV files.
  - Added support for setting the encryption domain on Check Point gateways (including clusters and dynamic IP gateways) and interoperable devices in CSV files.
  - Added support for setting the WebServer property, when configuring Hosts in CSV files.
  - Enhanced and corrected bugs in the support for rules from CSV files
  - Added support for a "modifications mode", used mainly to modify some properties on currently build objects in CSV files. This option is designed to be used with object dumper.
  - Enhanced support for NATted objects when importing from PIX configurations. Now the results are more accurate.
  - Enhanced support for Rules processing when importing from PIX and Cisco Router configurations: Now splits in different sections of the imported rulebase, the different ACLs that the configuration may have
  - Now, when creating groups from CSV Files the groups are created first and then elements are added to them.
- Object Dumper
  - Added support for services (TCP, UDP, ICMP, Other and RPC)
  - Added support to recognize basic rules from the rulebases (rulebases_5_0.fws) file.
  - Added support for interfaces with Check Point gateways (including clusters and dynamic IP gateways) and Interoperable devices. This includes interfaces with dynamic IP
  - Support for recognizing the webserver property on hosts
  - Support for recognizing the encryption domain on Check Point Gateways (including clusters and dynamic IP gateways) and Interoperable devices.
  - Enhanced the support for reading objects.C files from gateways.
  - Tabulation (-tab) mode is not supported anymore. (seems that nobody was actually using it)
- Documentation
  - For the first time, the tools have a more decent manual. It's the plan to enhance the documentation in the next releases

* Version 1.9.2 - November 2004

- Object Filler
  - Added support for TCP/UDP services when importing from CSV Files, Cisco PIX, Cisco Routers and NetScreen configuration files.
  - Added support for importing basic layer 3 and layer 4 rulebases from CSV files
  - Added support for importing rules from Cisco PIX and Cisco Routers with access-list configurations.
- Object Dumper:
  - Added support for objects.C files found on gateways, for recovery options.

- Added support for Check Point dynamic IP gateways.

* Version 1.8 - March 2004

- Object Filler
  - Support to DELETE and RENAME operations over objects, when specifying such operations on CSV files
  - Support to Domain and Dynamic objects.
  - Clusters and cluster members are also now supported.
  - Support to the following SmartLSM objects: IP40 ROBO gateways, Edge X ROBO gateways, profiles (SmartLSM VPN-1 Egde/Embedded NG profiles).
  - Support to PIX network-object groups and to name statements.
  - Support to Raptor configuration files.
  - Support to Cisco IOS ACLs (including the ones declared with inverse masks).
  - Support to NONAT mode when importing from files.
- Object Dumper
  - Added support to clusters and cluster members.
  - Added support to Domain objects.
  - Added support to SmartLSM VPN-1 Edge/Embedded profiles
  - Added support to Dynamic Objects.

* Version 1.6 - January 2004

- Object Filler
  - Fixed bug: Now it works with IP address range objects behind a NAT IP range.
  - Fixed bug: Now it recognizes "cpgws" as a valid object type when using command line parameters.
  - Fixed bug: Correctly handles objects hidden behind "All gateways", when "All" is specified on the NATting Object column, when importing from CSV files.
  - Enhanced interactive mode with more comments to ease user experience.
  - Improved support to NAT statements when importing from PIX.
  - Improved summary information when importing information from any file.
  - Now groups are supported when using the cgi mode (HTML form) and importing from CSV files.
  - Support to import from SideWinder configuration files.
  - New output mode (ASCII) introduced, which instead of writing dbedit commands, leaves the information on CSV format, which is easier to read and compare.
  - Support to Interoperable Devices, Plain Gateways and OSE Devices when importing from CSV files and when generating objects from command line.
  - Now when importing from a LIST type of file, it's possible to create ranges and groups.
  - Native binary support for Linux/SecurePlatform.
- Object Dumper
  - Now it recognizes Interoperable Devices, Plain Gateways and OSE Devices.
  - Enhanced interactive mode with more comments to ease user experience.
  - HTML mode for output files is now available.
  - TAB (Column) mode for output files is now available.
  - Native binary support for Linux/SecurePlatform.

* Version 1.4 - December 2003

- Support to interactive mode (command line is still supported).
- Enhanced support to duplicates (now it takes in account the object type, not only the IP address).
- IP ranges support when creating objects and when importing from CSV files.
- Support to import objects from Gauntlet configuration files.
- Support to comments when importing from CSV files and List files.

- Support to import groups when importing from CSV files.
- Support to groups when importing from CSV files.
- Gives a summary of how many objects of each known type were processed.
- **Object Dumper companion tool was created**.

* Version 1.2 - July 2003
  - Support to import objects from CSV (comma separated) files, where you can detail the objects you need to create.
  - Support to import from lists files, where you just detail IP and netmask,
  - Object name and everything else is calculated automatically.
  - Support to import objects from operating system's host file.
  - Support to import objects from Cisco PIX and NetScreen configuration files, and create network objects from there. Importing rules is not supported.
  - Support for Hide NAT on created sequential objects.
  - Support for Static NAT on imported objects from files.
  - Support to NAT ranges to hide created objects.
  - Support for color specification on new objects.

* Version 0.96 - May 2003
  - Support for importing from CSV files.
  - Support to Check Point Host and Check Point Gateway objects.
  - Support to Hide NAT for objects generated using command line.
  - Support for colors on created objects.

* Version 0.8 - April 2003
  - This is the Initial Public release ("First Customer Shipment").
  - Supported only creating hosts and networks.

* Version 0.5 - April 20th, 2003 – 06:57 AM
  - This idea of Object Filler is born. I started with some preliminary designs, algorithms and coding that morning.

Known limitations, issues and particular behavior for both programs

Object Filler's known limitations and issues

- Object Filler cannot detect if there's enough space on disk for the output file. So, please be careful and send program's output to a file placed on a disk or partition with enough space for this output. For each object created, take an average size of 750 bytes each. Usually it will take less space, but using this consideration you will be safe.
- When importing from Cisco PIX configurations, IP Address ranges specified on the original configuration won't be processed. If you've a range like 1.2.3.4-1.2.3.99, this range will simply be ignored. Individual IP's will be processed, however.
- When importing from Cisco PIX configurations, and there are 2 global statements (one for the external interface and one for the DMZ, as an example), only the first one found is applied and the second is ignored.
- When importing from Cisco PIX configurations, the use of names in access-list statements is not supported. The import of an access-list that contains names will result in a problematic rule.
- When importing from Cisco PIX configurations, the import of service groups is not supported.
the use of names in access-list statements is not supported. The import of an access-list that contains names will result in a problematic rule.
- When importing from NetScreen configurations, Hide NAT and PAT are not supported, so if you have "dip" or "vip" statements, they will be ignored as NAT statements but will be processed as any other normal line (i.e. the IP addresses will be converted to objects) . Please note that if you have "mip" statements (Static NAT), they will be processed as Static NATted objects, and the right output will be produced.
- When importing from NetScreen configurations, the import of service groups is not supported.
- When importing from SideWinder configurations, domains and service groups are not supported while working over the ACL tables.
- 0.0.0.0 is not a valid IP address for Object Filler.
- When importing from files, due file internal representation, sometimes the last line of the file will be processed twice. However, the output for this line will be produced just right (once), since Object Filler detects and avoids duplicates by default.
- When importing groups from CSV files, it's not possible to specify color nor comments for them. This is planned to be fixed, but currently not supported
- It's not possible to include groups as members of other groups, when importing LIST files with Object Filler.
- OSE devices cannot have NAT properties defined.
- It's not possible to use Check Point Hosts, Check Point Gateways, Check Point Dynamic Gateways, OSE Devices, Interoperable Devices, Plain Gateways or Services, when importing LIST files with Object Filler.
- When defining interfaces, the interfaces have to be defined after the gateways that owns such interfaces. Also, Object Filler assumes that it's defining the first interfaces on the object. Adding interfaces to an object that already has interfaces defined is not supported.
- When creating rules from CSV Files, adding rules to an existing policy is not supported.
- When processing rules from CSV Files, using User Groups as sources (user@location) is not supported
- When processing rules from CSV Files, using resources for services (uri resources, smtp resources, etc.) is not supported

- When Object Filler is processing rules, only Accept, Drop and Reject are accepted as valid actions. Other actions are not supported
- The tool has been not tested with VSX management.

Object Dumper's known limitations and issues

- Only the following object types are supported: Check Point Hosts, Check Point gateways, Check Point Dynamic Gateways, Check Point Clusters, Check Point Cluster Members, plain hosts, networks, IP Address ranges, OSE Devices, Plain Gateways, OSE Devices, Domains, Dynnamic Objects, Interfaces and Groups. TCP, UDP, ICMP, RPC, DCE-RPC and type Other Services are supported. Other network object types and services are currently not supported.
- Object Dumper does not support SmartLSM VPN-1 Edge/Embedded ROBO gateways, like IP40 or Edge X SmartLSM ROBO gateways.
- Comments and colors for groups are not fully supported.
- Currently, if you want to process a rulebases file, you must process an objects file also. Processing just the rulebases file by itself is not currently supported.
- When processing rules from rulebases_5_0.fws, using User Groups as sources (user@location) is not supported. Instead, the name of the network (the location part of the source) is showed.
- When processing rules from rulebases_5_0.fws, the resources for services (uri resources, smtp resources, etc.) are not recognized properly.
- When processing rules from rulebases_5_0.fws, only Accept, Drop and Reject are accepted as valid actions. Other actions are not supported.
- When processing rulebases, if the rulebases_5_0.fws contains several policy packages (i.e. policies with different names), these are exported as rule section headers (not different rulebases), and recognized by Object Filler as sections, not as different policies.
- When exporting objects with Object Dumper, due some limitations on the program, some times the color may be exported wrongly. On these cases, the object's color will be reset to black.
- Object Dumper was not tested on this version with Provider-1.
- The tool has been not tested with VSX management.

Tested scenarios

These programs are not official by Check Point Software Technologies, nor supported in any way by any entity.

These programs can run in a native form and have been tested by the author and others on:

- Microsoft Windows NT 4.0 and Microsoft Windows 2000.
- Red Hat Linux 7.2
- SecurePlatform Next Generation with Application Intelligence R55 (NG+AI R55)

However, since program's output is just regular ASCII TEXT, it can be easily transferred (using scp, ftp, diskettes or any other file transfer mechanism) to a SmartCenter Server on other operating system, as shown below. Please **make sure** that if you transfer files among operating systems you do that as ASCII TEXT files.

Since the tools are programmed under windows, they are tested more deeply in this platform. Then they are recompiled under GNU/Linux and tested for basic functionality. While this means not so much testing on GNU/Linux, the functionality should be the same. I recently got report of somethings working in Windows but not in Linux. If it happens that you find something like this, I'd appreciate very much your report on it.

Requests for natively supporting other OSes by the programs are always welcome and taken in account. Solaris SPARC is in the roadmap for future releases, but not any firm dates yet.

Object Filler's output has been tested so far with SmartCenter Servers running on:

- SecurePlatform: NG+AI R55
- Nokia IPSO: NG+AI R55
- Windows NT 4.0: NG FP3
- Windows 2000 Server: NG+AI R54, NG+AI R55

Object Filler's output has also been tested with Provider-1 NG+AI R54, MDS Manager and Container Server, under Solaris 2.8

To review what specific software/firmware versions were tested when importing configurations from other brands, please take a look on the proper section below on this document.

Object Dumper has been tested using Objects_5_0.C files coming from:

- SecurePlatform: NG+AI R54, NG+AI R55
- Nokia IPSO: NG FP3, NG+AI R54
- Windows: NG FP3, NG+AI R54, NG+AI R55

As you may imagine, my testing resources are finite and small, so if you use this program on a different environment, I'd really appreciate if you send me a note saying so, will surely help improving this documentation, and everybody that uses these programs.

In the event of any program bug, failure, request, comment, grammar correction on this documentation or the messages sent by the programs, or any other request for enhancement, **PLEASE** send an e-mail to ***mhoz@mexico.com*** with the subject *About Object Filler*. I don't promise I'll fix immediately whatever you're asking for, as I'm doing this on my -not so abundant- free time, but I'd like to hear from you anyways, and I promise to try to implement your suggestion, whatever that is. In general, so far I've been able to implement specific features, in a two-week timeframe, for people asking/requesting me such features and willing to help on testing them.

If you wish, you may use the provided public PGP key (included in the distribution file) to encrypt any files or message directed to the author.

Always keep in mind also that this is a not official nor supported software.

Tools Installation

The tools do not need to be installed at all. They are executables that don't need any special library. Under Un*x flavors, the tools need access only to the standard libraries that any other program needs.

In the other hand, remember that you don't have to have the executable running natively on the platform where you have the SmartCenter server. If you have the SmartCenter Server on Nokia IPSO for example, you could execute Object Filler or Object Dumper in any other platform, by just transferring the needed files via FTP, SCP or any other file transfer mechanism.

Introduction to the object types used by the tools

Introduction

All the elements from which the a network security policy are represented by *Objects* in SmartCenter. Each object is an atomic element that has different properties. You may see an introduction of this in the SmartCenter manual that comes on the CD where the Check Point software is distributed. The Chapter 1 (SmartCenter Overview) contains a section named "Managing Objects in SmartDashboard", as well as the Appendix A named "Network Objects" where you may find more information with regards to Objects.

The following is a brief and simple explanation on common situations for the objects, and what to expect or when to use them while using Object Dumper and/or Object Filler.

Network Objects Definition

Check Point Hosts
Represent servers with one or more NICs (interfaces) attached to them, but where no routing through it is performed (packets cannot go from one interface to another). If a Check Point Host has more than one interface, all of them will be automatically marked as external. Check Point Hosts usually indicate VPN-1/FireWall-1 SecureServers or SmartCenter Servers that are in distributed configuration.

Check Point Gateways
Represent gateways - i.e. hosts with more than one interface, but where the packets are processed, routed and passed (if allowed) between interfaces. Usually indicate Check Point Gateways (Enforcement Points), either StandAlone or distributed configuration.

Check Point Dynamic Gateways
These are Check Point Gateways, where the main IP address is a dynamic IP Address (i.e. is not a Fixed IP Address)

Check Point Cluster
A group of Check Point Gateways that behave as if they were just one entity.

Check Point Cluster Member
A Check Point Gateway that belongs to a cluster.

Plain Host
A host with one or more interfaces, where no Check Point product is installed, and where no routing among interfaces is performed. Used to represent user's machines, workstations, hosts or servers.

Network
A simple network segment, delimited by an IP address and a netmask.

Plain Gateway
A device that passed packets through it – i.e. performs routing among interfaces, but has no Check Point products installed.

Interoperable Device
A Plain Gateway that has some sort of VPN software installed on it, and has the capability of doing VPN with a Check Point Gateway.

OSE Device
Is a Cisco, Nortel or 3Com device from which is possible to read and/or write security rules.

IP Address range
A group of IP addresses that cannot be delimited with a netmask, and that for the effects of a security policy, behave like one.

Dynamic Object
An object that takes different IPs, depending of the associations made at the gateway level, once the security policy has been applied.

Domain
A domain name.

Network Object Group (Simple Group)
A set of Network objects.

Interface
An interface that belongs to a gateway.

SmartLSM Profile for NG Embedded
A Profile going to be used to define security policies and VPNs with NG Embedded devices, with Smart Large Scale Manager (LSM). Valid only if SmartLSM has been enabled on the SmartCenter.

SmartLSM Edge X gateway
A gateway defined as Check Point VPN-1 Edge X gateway, usable with SmartLSM. Valid only if SmartLSM has been enabled on the SmartCenter.

SmartLSM IP40 gateway
A gateway defined as Nokia IP40 gateway, usable with SmartLSM. Valid only if SmartLSM has been enabled on the SmartCenter.

Services Definition

TCP Service
Service definition that uses the Transmission Control Protocol (TCP). The main property for this is a Port number that may go from 1 to 65,535

UDP Service
Service definition that uses the User Datagram Control Protocol (UCP). The main property for this is a Port number that may go from 1 to 65,535

ICMP Service
Service definition that uses the Internet Control Message Protocol (ICMP). The main property is the ICMP type, as defined in several RFCs.

RPC Service
It is a definition that makes reference to service running over Remote Procedure Calls. The definition of the service is made with program numbers.

DCE-RPC Service
Distributed Computing Environment/Remote Procedure Call. It's a different kind of RCP services. The identification is made using UUIDs.

Table of Objects, Services and Operations supported by Object Filler

	CLI mode	CSV File create	CSV File modify	Supports NAT	Webserver property	Encryption Domain
Network Objects						
Check Point Host	ss	ss	modss	Yes	No	No
Check Point Gateway	cpgw	cpgw	modcpgw	Yes	No	Yes
Check Point Dynamic Gateway	N/S	dynamicgw	moddynamicgw	No	No	Yes
Check Point Cluster	N/S	cluster	modcluster	No	No	Yes
Check Point Cluster Member	N/S	member	N/S	No	No	No
Plain Host	host	host	modhost	Yes	Yes	No
Network	net	net	modnet	Yes	No	No
Plain Gateway	plaingw	plaingw	modplaingw	Yes	No	Yes
Interoperable device	idevice	idevice	modidevice	Yes	No	Yes
OSE Device	ose	ose	modose	No	No	No
IP Address Range	range	range	modrange	Yes	No	No
Dynamic object	N/S	dynamic	N/S	No	No	No
Domain	N/S	domain	N/S	No	No	No
Network Objects Group (Simple)	N/S	group	N/S	No	No	No
Interface	N/S	interface	N/S	No	No	No
SmartLSM Profile for Embedded NG	N/S	lprofile	N/S	No	No	No
SmartLSM VPN-1 Edge X gateway	ledge	ledge	N/S	No	No	No
SmartLSM IP40 gateway	lip40	lip40	N/S	No	No	No
Services						
TCP Service	N/S	tcp	modtcp	N/A	N/A	N/A
UDP Service	N/S	udp	modudp	N/A	N/A	N/A
ICMP Service	N/S	icmp	N/S	N/A	N/A	N/A
RPC Service	N/S	rpc	N/S	N/A	N/A	N/A
DCE-RPC Service	N/S	dcerpc	N/S	N/A	N/A	N/A
Other Service	N/S	other	N/S	N/A	N/A	N/A
Services Group	N/S	srvgroup	N/S	N/A	N/A	N/A
Operations						
Delete operation	N/S	DELETE	N/A	N/A	N/A	N/A
Rename operation	N/S	RENAME	N/A	N/A	N/A	N/A

N/S=Not Supported. N/A=Doesn't apply
* CLI mode means it is supported by Object Filler from Command Line. The content of the cell is the keword used.
* CSV File create means that object can be created via a CSV File.
* CSV File modify means that the object exist, but its properties will be modified.
* WebServer property means that the object will be marked as webserver
* Encryption domain means that the object will have an encryption domain associated.
* Operations Delete and Rename are only supported for Network Objects, not services.

Table of Objects, Services and Operations supported by Object Dumper

	Supported	Interfaces	Encryption Domain
Network Objects			
Check Point Host	ss	No	No
Check Point Gateway	cpgw	Yes	Yes
Check Point Dynamic Gateway	dynamicgw	Yes	Yes
Check Point Cluster	cluster	No	Yes
Check Point Cluster Member	member	No	No
Plain Host	host	No	No
Network	net	No	No
Plain Gateway	plaingw	Yes	Yes
Interoperable device	idevice	Yes	Yes
OSE Device	ose	No	No
IP Address Range	range	No	No
Dynamic object	dynamic	No	No
Domain	domain	No	No
Network Objects Group (Simple)	group	No	No
Interface	interface	No	No
SmartLSM Profile for Embedded NG	lprofile	No	No
SmartLSM VPN-1 Edge X gateway	N/S	No	No
SmartLSM IP40 gateway	N/S	No	No
Services			
TCP Service	tcp	N/A	N/A
UDP Service	udp	N/A	N/A
ICMP Service	icmp	N/A	N/A
RPC Service	rpc	N/A	N/A
DCE-RPC Service	dcerpc	N/A	N/A
Other Service	other	N/A	N/A
Services Group	srvgroup	N/A	N/A

N/S=Not Supported. N/A=Doesn't apply
* Supported means that Object Dumper will recognize the object, and the output will have the keyword listed.
* Interfaces means that if the object has interfaces attached, such interfaces will be listed in the output.
* Encryption domain means that if the object has an encryption domain associated, it will be listed in the output.

Object Filler

Program syntax

```
1) ofiller  help  (prints help pages - with examples)
2) ofiller  -f file -i input [-o|-a] file [-c color] [-t type]
            [-p policy] [-nopv] [-nonat] [-v]
3) ofiller  -t type -s ip -d ip -m mask [-c color]
            [-n ip | -ns ip -nd ip -nm mask] [-b obj] [-o|-a] file [-v]
```

Program syntax #1 : Asking for help

```
ofiller help
```

This syntax allows you to see the incorporated help in the program. The result is simply a brief documentation on the program's switches

Program syntax #2 : Importing from a file

```
ofiller  -f file -i input [-o|-a] file [-c color] [-t type]
         [-p policy] [-nopv] [-nonat] [-v]
```

This syntax allows you to produce objects information (and possibly rules information) having as source for it, a file. This file may be a CSV File in a pre-defined format, or the configuration of another firewall.

-i - Input type - It can be either:

* csv - File must be formatted on csv format. You may take a look on the file sample_csv.csv for more information on this switch.

* list - File must contain 2 mandatory fields: IP Address and netmask. You may take a look on the file sample_list1.csv for more information.

* hosts - File has the format of a hosts file (/etc/hosts on Un*x systems, or %SYSTEMROOT%\\system32\\drivers\\etc\\hosts on Microsoft Windows systems). You may take a look on the sample file sample_hosts for more information.

* pix - File is the configuration listing from a Cisco PIX device. You can get this information from a PIX device using the command \"show running\" or \"write terminal\".

* netscreen - File is the configuration listing from a NetScreen device. You can get this information from a NetScreen device using the command \"get config all\".

* gauntlet - File is the configuration file of Gauntlet (gauntlet.conf).

* sidewinder - File is the configuration file of SideWinder (ACL and ipfilter files).

* raptor - File is the Raptor configuration file that contains the IPs and rules of the firewall.

* ciscorouter - File contains is the result of executing the \"show running\" command from a device running Cisco's IOS.

This is a required parameter.

Example:

```
ofiller -i csv -o all_objects.txt -f input.txt
```

-f - input File - Takes the input from the specified file. See details on –i switch on how this file needs to be formatted. **Required parameter.**

Example:
```
ofiller -f my_old_5XT.cfg -o output.dbedit -i netscreen
```

-p - Policy name - It specifies the policy name that the imported policy will have when it's imported into the SmartCenter. It also used to tell Object Filler that you want to import a policy. If you don't specify this switch, even if the configuration contains a policy, Object Filler won't try to process it. This switch is only valid for the supported input files, currently Cisco PIX, Cisco Routers and CSV Files. **Optional parameter**.

Example:
```
ofiller -p mypolicy -f Conf_PIX515.txt -o output.dbedit -i pix
```

-nopv - No Policy Verification - This switch is used to decide if the objects will be verified to see if they were processed by Object Filler before or not. This switch is relevant only if used with -p switch. If it is not specified, all the objects present in rules definition that have not been processed, will be translated to \"Any\". Use this switch if you are processing only rules, but no objects. **Optional Parameter.**

Example:
```
ofiller -nopv -p mypolicy -f new_rules.csv -o output.dbedit -i csv
```

-c - Color - The color we'll use to build the objects. Can be black blue, green, gray, red, pink, brown, cyan, yellow, orange, magenta, sienna, gold, coral, firebrick. When importing from a File, this parameter will take precedence over any specified color on the importing file. **Optional parameter.**

Example:
```
ofiller -c blue -f c:\files\my_old_535.txt -i pix -o d:\\tmp\\objects.txt
```

-t - object Type - It's the object type we'll build can be host, cpgw (Check Point Gateway), ss (SecureServers - Check Point Host), idevice (Interoperable Device), plaingw (Plain Gateway) or ose (OSE Device). This parameter will be relevant only if importing from a hosts file. If specified with any other type of file, it will be ignored**. Optional parameter**.

Example:
ofiller -t ss -f /etc/hosts -i hosts -o /home/admins/root/host_smc.txt

-o - Output file - The name of the file where resulting dbedit commands will be stored. Please make sure you have enough disk space to store all produced commands. To calculate this pace, take an average of 750 bytes per object to process. The File must not exist previously.. If it exists, the execution of the program will be aborted. This switch is mutually exclusive with -a (i.e. if you can only specify -a or -o, but must at least use one of them). **Required parameter if –a was not specified.**

Example:
```
ofiller -o all_objects.dbedit -i csv -f input.txt
```

-a - Ascii file - The name of the file where resulting CSV information will be stored. This is an alternative to -a, and instead of writing dbedit commands, Object Filler writes information regarding the created objects (name, ip, comments, etc.) on CSV format, so you can take a look on a spreadsheet program first. Please make sure you have enough disk space to store all produced commands. To calculate this space, take an average of 120 bytes per object to process. File must not exist previously. This switch is mutually exclusive with -o (i.e. if you can

only specify -a or -o, bust must at least use one of them). **Required parameter if –o was not specified.**

Example:
```
ofiller -a all_objects.csv -i netscreen -f ns5200.conf
```

-nonat - Use this option if you want that the importing file NAT statements not to be processed. This will cause that build objects won't be NATTEd, even if they are on the original configuration used to feed Object Filler. **Optional parameter.**

Example:
```
ofiller -a all_objects.csv -i netscreen -f ns5200.conf -nonat
```

-v - Verbose mode - shows on the console (the screen) details on how the processing is being done line-by-line. This is very useful especially when importing files, since it says how each line was treated. **Optional parameter.**

Example:
```
ofiller -v -i csv -o all_objects.txt -f input.txt
```

Program syntax #3 : Specifying arguments from command line

```
ofiller  -t type -s ip -d ip -m mask [-c color]
         [-n ip | -ns ip -nd ip -nm mask] [-b obj] [-o|-a] file [-v]
```

-t - object Type - It's the object type we'll build. It can be:

- cpgw (Check Point Gateways)
- ss (Check Point Hosts)
- host (Plain hosts)
- plaingw (Plain Gateways)
- net (Plain Networks)
- range (IP Address ranges)
- ose (OSE Devices)
- idevice (Interoperable devices)
- ledge (SmartLSM VPN-1 Edge X gateways)
- lip40 (SmartLSM IP40 gateways)

For the last 2 type of objects, please see the notes at the end of this document section.

The object type is a **Required parameter**.

Example:
```
ofiller -t host -s 10.2.3.4 -d 10.2.3.99 -m 24 -o output.dbedit
```

-s - Source ip - Indicates the first IP we'll use to build the ranges. Note that when building SmartLSM Edge X or IP40 gateways, this initial IP cannot be smaller than 0.0.0.10 (and it's recommended to be 0.0.0.10). **Required parameter**.

Example:
```
ofiller -s 10.2.3.4 -t net -d 10.2.30.99 -m 24 -o output.dbedit
```

-d - Destination ip - Indicates the IP where the range finishes. It must be "bigger" (network-wise) than Source IP. Note than when building SmartLSM Edge X or IP40 gateways, this ending IP cannot be bigger than 0.0.254.254. **Required parameter**.

Example:

```
ofiller -d 10.2.30.99 -t net -s 10.2.3.4 -m 24 -o output.dbedit
```

-m - Mask - The mask that we'll use to build the objects. Must be between 8 and 30 bits. Required parameter.

00 bits = 0.0.0.0	08 bits = 255.0.0.0
09 bits = 255.128.0.0	10 bits = 255.192.0.0
11 bits = 255.224.0.0	12 bits = 255.240.0.0
13 bits = 255.248.0.0	14 bits = 255.252.0.0
15 bits = 255.254.0.0	16 bits = 255.255.0.0
17 bits = 255.255.128.0	18 bits = 255.255.192.0
19 bits = 255.255.224.0	20 bits = 255.255.240.0
21 bits = 255.255.248.0	22 bits = 255.255.252.0
23 bits = 255.255.254.0	24 bits = 255.255.255.0
25 bits = 255.255.255.128	26 bits = 255.255.255.192
27 bits = 255.255.255.224	28 bits = 255.255.255.240
29 bits = 255.255.255.248	30 bits = 255.255.255.252
32 bits = 255.255.255.255	

Example:

```
ofiller -m 25 -t net -s 10.2.3.0 -d 10.2.30.0 -o net.dbedit
```

-c - Color – The color we'll use to build the objects. See Appendix B for a list of valid Colors. **Optional parameter.**

Example:

```
ofiller -c sienna -m 25 -t net -s 10.2.3.0 -d 10.2.30.0 -o net.dbedit
```

-n - NAT ip - The IP behind which the objects will be automatically NATted. If not specified, no NAT will be done to created objects. Only Hide NAT is supported on this syntax. It cannot be used with -ns, -nd and -nm switches. **Optional parameter**.

Example:

```
ofiller -n 192.168.1.3 -m 25 -t net -s 10.2.3.0 -d 10.2.9.0 -o n.txt
```

-ns, nd, nm - NAT range Starting ip, NAT range Destination ip, NAT range Mask - The IP address range behind which the created objects will be Hide NATted. Sometimes there is a big network (let's say a Class B network) with internal addressing, and then a C Class network with valid addresses. These switches allow the administrator to bind every created object to a different IP from a declared valid network. This way, if we have the 10.10.0.0/16 invalid network, and then the 172.16.200.0/24 valid segment, we can use Object Filler to automatically create objects like 10.10.0.0/24 NATted behind 172.16.200.1, then 10.10.1.0/24 NATted behind 172.16.200.2, next 10.10.2.0/24 NATted behind 172.16.200.3 and so on. When 172.16.200.254 (the lastIP of the valid range) is reached, then the next object will be NATted using 172.16.200.1 (the first IP in the NATting range) again.. – **Optional parameters**

Example:

```
ofiller -ns 192.168.200.0 -nd 192.168.201.255 -nm 24 -s 10.10.0.0 -d
10.20.255.255 -m 24 -t net -o nets.dbedit
```

-b - hiding oBject - The name of the Check Point gateway object which will NAT hide the created objects. Optional parameter, but when specified -n must be also used. This object must exist on the SmartCenter before you attempt to use this switch, and the name of the object in the SmartCenter must be exactly as specified here, as all involved programs (including ofiller and dbedit) are case sensitive. If -n was given, but -b not specified, then objects will hide behind All gateways (*All) as default. **Optional parameter.**

Example:

```
   ofiller -b The_Wall -n 10.9.8.7 -m 25 -t net -s 10.2.3.0 -d 10.20.9.0 -o
x.txt
```

-o - Output file - The name of the file where resulting dbedit commands will be stored. Please make sure you have enough disk space to store all produced commands. To calculate this pace, take an average of 750 bytes per object to process. The File must not exist previously.. If it exists, the execution of the program will be aborted. This switch is mutually exclusive with -a (i.e. if you can only specify -a or -o, but must at least use one of them). **Required parameter if –a was not specified.**

Example:
```
ofiller -o dbedit_commands.txt -s 10.2.3.0 -d 10.2.3.9 -m 25 -t host
```

-a - Ascii file - The name of the file where resulting CSV information will be stored. This is an alternative to -a, and instead of writing dbedit commands, Object Filler writes information regarding the created objects (name, ip, comments, etc.) on CSV format, so you can take a look on a spreadsheet program first. Please make sure you have enough disk space to store all produced commands. To calculate this space, take an average of 120 bytes per object to process. File must not exist previously. This switch is mutually exclusive with -o (i.e. if you can only specify -a or -o, bust must at least use one of them). **Required parameter if –o was not specified.**

Example:
```
ofiller -a preview.csv -s 10.2.3.0 -d 10.2.3.9 -m 25 -t host
```

-v - Verbose mode - shows on the console (the screen) details on how the processing is being done line-by-line. This is very useful especially when importing files, since it says how each line was treated. **Optional parameter.**

Example:
```
ofiller –v -o dbedit_commands.txt -s 10.2.3.0 -d 10.2.3.9 -m 25 -t host
```

Building SmartLSM Objects with command line

When building SmartLSM ROBO gateways (ledge or lip40 object types), there are some rules that apply, and you must know:
- Object Filler assumes there are no previously created SmartLSM or regular Edge/Embedded NG objects previously created, nor any kind of profiles, nor any dynamic objects
- The first IP cannot be lower than 0.0.0.10
- The last IP cannot be higher than 0.0.254.254
- Automatically creates a SmartLSM VPN-1 Edge/Embedded NG profile called gen_profile with IP 0.0.0.8, used as the default profile on the created ROBO gateways.
- Automatically creates a Dynamic Object called gen_dyn_obj with IP 0.0.0.9 used on the created ROBO gateways.
- Automatically assigns the IP Address range 1.2.3.4-1.2.3.5 to all created objects.
- It doesn't assign any registration key for the created ROBO gateways.

Important note regarding IP addresses for SmartLSM related objects from command line
When building SmartLSM ROBO gateways with the command line, specified IP Addresses for the objects must be in the range from 0.0.0.10 to 0.0.254.254. Please note that you *must make sure* that no duplicate IPaddress exist on the configuration. To assure this, please log in to the SmartLSM GUI, sort the elements by "ID" (second column from left to right) and make sure the IPs you are specifying are not listed there. Then, use Object Dumper to dump the contents of your current Objects_5_0.C file and see that no profile or dynamic object is already using the IPs you're trying to assign.

To avoid any problems or IP conflicts in any case, is **highly** recommended to have the SmartCenter Server clean of SmarLSM objects, dynamic objects, profiles or any kind of dynamic objects. This is, you should use Object Filler just for the initial configuration, unless you know what you're doing.

The usual recommendation when building SmartLSM VPN-1/Embedded ROBO gateways using the command line, is to direct the output to a CSV file using the -a option of Object Filler:

```
ofiller -t lip40 -s 0.0.0.10 -d 0.0.0.210 -m 24 -a output.csv
```

Then edit the resulting file (output.csv on this case) to fill it with the proper ROBO gateway information. Finally, use the -i csv option to build the dbedit commands:

```
ofiller -f output.csv -i csv -o robogws.dbedit
```

This way, you can greatly automate the building of the new ROBO gateways.

Examples of program execution

```
ofiller -f source.csv -i csv -o objects.txt
```
Will take data from file source.csv , with CSV format and leave results (dbedit commands) on a file named objects.txt

```
ofiller -i csv -f source.csv -o objects.txt -v > results.txt
```
Same as above, but now the program's verbose output will be directed to a file named *results.txt*, instead of the console, so you may review it later.

```
ofiller -t hosts -s 10.0.0.0 -d 10.100.0.0 -m 24 -o hosts.dbedit
```
Will build hosts from 10.0.0.0 to 10.100.0.0 skipping network addresses as well as broadcast addresses, using 24 bits as objects netmask. Output will be directed to hosts.dbedit file

```
ofiller -t nets -s 2.0.0.0 -d 2.5.6.2 -m 24 -c blue -n 1.2.3.4 -b FireWall -o
nets.txt
```
Will build networks from 2.0.0.0 to 2.5.6.2 skipping broadcast addresses, using 24 bits as objects' netmask. Objects will be created on color blue, Hide NATed behind 1.2.3.4 and an object named FireWall. This firewall gateway name must be exactly as specified here, as program is case sensitive.

```
ofiller -f 535.conf -o imp_pix.csv -i pix -v
```
Will import a PIX configuration from a file named 535.conf and leave the output (CSV formatted) in the file imp_pix.csv

Importing configurations from files with Object Filler

Importing files in general

First thing you have to know is that Object Filler and Object Dumper don't need dos2unix conversions on ***input files***. This is, if you get a file from a Solaris or Nokia IPSO machine, you can get it to Windows, and it doesn't matter the format you transfer it with, will be processed the right way. However, for the ***output files*** may need dos2unix conversions if you move files to a different machine from which they were generated. If you're transferring a dbedit command file (the results from running Object Filler) over FTP, you must transfer it as ASCII, not Binary file.

Always try to use the ASCII output mode (-a switch) to review what Object Filler whould do, review the results, and then run Object Filler again on the original file, but using the dbedit mode (-o switch) to finally produce the dbedit commands you'll use to import into the SmartCenter.

If when you run Object Filler you don't choose the right file type (i.e. you ask to translate a PIX configuration, whereas you have in the file a NetScreen configuration), Object Filler will try to figure that out and will tell this suspicious status, but don't rely on this mechanism and always try to specify the right type of file.

Please be aware that if you have already an object that has the same name of an object you're importing, the only property of the object that will be modified is the IP Address, and if the object is also NATted, the NATting properties will be modified too...

In general, if you are going to populate a SmartCenter that already has data on it, it is strongly recommended to export your current list of objects, and compare it to the one that will be imported, may be using the ASCII output mode (-a) of Object Filler, and using Object Dumper to export your current configuration. This way you will notice which object have chances to be modified before you do any changes to your live configuration.

If possible, it is recommended to use Object Filler only to populate empty SmartCenter Servers.

Comma Separated Values (CSV) file type

This is by far the most powerful (but also the more complex) file format supported by Object Filler.

File must be formatted on CSV format, i.e. all values must be separated by a comma. The only special consideration is that columns order must be preserved as declared on the sample file and as is explained below. Non-used spaces can be just left empty (or filled with zeroes), but the space still has to be defined by a comma however.

Format of the CSV File used by Object Filler

This is the definition of the CSV file format used by Object Filler to take information to build or modify objects from. Also, it is the format on which Object Dumper leaves the information after processing the input files specified for it.

- Column 1 – *Object Name*
The name the object will have. Please consider the naming conventions for objects on SmartCenter. Usually the more important things to remember here are: No spaces are allowed (use dashes and underscores instead), names must start with a letter (no numbers), and limit the names shorter than 32 characters.

- Column 2 - *Type of object* or *operation*.
Use the same as the supported object types on Object Filler with command line, or the ones listed in the table with supported types above: *ss*, *cpfw, dynamicgw, cluster, member, host, net, plaingw, idevice, ose, range, dynamic, domain, interface, lprofile, ledge, lip40, tcp, udp, icmp, other, rpc, dcerpc*. Please see documentation below for building groups.

When creating interfaces, it's important that the interface is defined after the gateway that owns such interface is define. If you define it before, the creation of such interface will not complain on Object Filler, but at the import time with dbedit, it will fail.

If you are changing the properties of an object, it's also accepted on this column to have specified *modss, modcpgw, moddynamicgw, modcluster, modhost, modnet, modplaingw, modidevice, modose, modrange, modtcp, modudp.*

The type field can also be used to specify an operation. Currently RENAME (to change the name of a network object) and DELETE (to delete a network object) operations are supported. Provided object names must match the case of the real object names. Objects are not verified that they were processed by Object Filler before, nor that they currently exist on the SmartCenter.

- Column 3 - *IP Address*, *Initial IP Address*, *Port Number*.
This column usually contains the object's main IP address in "dotted" format like 1.2.3.4 – In the case of IP Address ranges, this column contains the initial IP of the range. In the case of TCP or UDP Services, it contains the port number, which can have a preceding > or < sign.

- Column 4 – *Netmask, Final IP Address, Timeout*
This column regularly specified the netmask in "dotted" format like 255.255.0.0 or 255.255.252.0. However, in the case of IP Address ranges, this column contains the final IP address of the range. In the case of TCP or UDP services, it contains the timeout for the service, which can be either "default" (the default global timeout specified in the SmartCenter), or a number in seconds.

- Column 5 – *Color* or *Owner device*
This column contains the color of the object. For a list of valid colors, please see Appendix B. When no valid color is specified, then black is assumed.
Howerver, if the object being specified is an interface, then this column contains the name of the object that owns such defined interface

- Column 6 – *NATting IP*, *Interface location*, *replies accepted*.
This column contains the NATting IP behind which the object will be NATted. This is optional.
When the object being processed is an interface, however, this column contains the interface location (internal or external). If the object is a service in the other hand, then it specifies if the service accepts replies or not. If they are accepted, then the column should list the word *replies*

- Column 7 – *NATting object*, *Interface Topology*
The name of the Check Point Gateway behind which the object hides. This is especially useful when the same SmartCenter is managing several Check Point firewalled gateways, and you want to perform NAT using only one of them. If not specified (if the column is empty, but NATting IP has being specified), or if *All* is used, then it will hide behind **All* the gateways managed by the SmartCenter.
In the case of Interfaces, this column specifies the IP addresses behind this interface (the topology). The valid values are *undefined*, which means there is no topology defined; *local,* which means all the IP addresses in the network specified by the interface's IP address; or the name of a network object (this has to be of type *network* or *network object group*) to be defined as the specific topology information for this interface.

- Column 8 - NAT type
It's the type of NAT that will be used for this object. Accepted values for this column are *Static* and *Hide*. If empty, but a NATting IP has been defined, Hide NAT type will be used by default.

- Column 9 - *Comments*
This column is used by all objects to put comments.

- Column 10 – Additional properties (*webserver*, *encdomain*)
When processing plain hosts, this column may contain the keyword *webserver*, which means that this object will be marked as a webserver for the effects of SmartDefense settings.
When processing Check Point Gateways, Check Point Dynamic Gateways, Check Point Clusters, Plain Gateways or Interoperable Devices, this column may contain the keywork *encdomain,* to specify that a manually defined encryption domain will be defined for this object

- Column 11 – *Protecting gateways*, *Encryption domain*
When processing plain hosts, and the host has been defined as webserver by the previous column, this column may specify behind which Check Point gateways this webserver is protected. If *All* is specified, it will be enforced behind all the gateways. If a gateway name is specified, this gateway will be the one specified as the protecting one. If leaved blank but webserver was specified in column 10, *All* will be assumed
When processing Check Point Gateways, Check Point Dynamic Gateways, Check Point Clusters, Plain Gateways or Interoperable Devices, this column contains the object (network or network object plain group) that will be used as the encryption domain, if the keyword *encdomain* was specified in column 10

Following are a couple of examples:

```
MyServer, host, 1.1.2.8,255.255.255.255, blue,          ,     ,     , HTTP Srvr
Users,    net,  1.2.0.0,255.255.0.0,     green,10.1.1.1, FW_3, Hide, users net
GW1,      cpgw,10.3.3.1,255.255.255.255,  black,        ,     ,      ,Main FW
eth0,     interface,10.3.3.1,255.255.255.0, GW1, external,    ,      ,
eth0,     interface,1.1.2.1,255.255.255.0, GW1, internal,Users,      ,
```

The first line will create a host named "MyServer" with IP 1.1.2.8, color blue and will have "HTTP Srvr" as comment.

The second line will create a network named "Users" with IP 1.2.0.0 and netmask 255.255.0.0 which will be Hide NATted behind the IP 10.1.1.1 and the Check Point gateway FW_3. In the comments field we'll have "users net" as comment. Object will be of color green.

The third, fourth and fifth lines define a Check Point Gateway with interfaces that belong to it.

When defining IP address ranges, you must define two IPs: the startingIP on the "IP address" column, and the ending IP of the range on the Netmask column (column 4). The ending IP must be "greater" network wise than starting IP, or Object Filler will reject it. The following is a valid example:

```
Int_Srvrs, range, 1.2.3.50, 1.2.3.60, green, 10.1.1.1, GatewayA, Hide, servers
```

This line will build an IP address range objects named "Int_Srvrs" from the IP 1.2.3.50 to IP 1.2.3.60, with green color, NAT Hide behind IP 10.1.1.1 and behind Check Point Gateway (which must be previously defined) GatewayA, and will use "servers" for the comment field.
When modifying properties, you may use the *mod* object types:

```
MyServer, modhost,1.1.2.8,255.255.255.255,blue , , , ,HTTP Srvr,websever,GW1
GW1,      modcpgw,1.1.2.1,255.255.255.255,black, , , ,Main FW,  encdomain,users
```

The lines above will modify a currently existing MyServer host object, will mark it as webserver for SmartDefense purposes, protected by gateway GW1
The second like will modify the already existing GW1 Check Point gateway, and will define the network *users* as the encryption domain for it.

CSV file type and Service objects

Since Object Filler 1.9.2 you can define TCP and UDP services using CSV files. Since Object Filler 2.0 ICMP, RPC, DCE-RCP and Other Services are also supported.

The format you must follow is this: *name, type, number, timeout, color, replies, expression*

name is the name you will give to the service.
type can be *tcp, udp, icmp, other, rpc* or *dce-rpc*
number is the port number that will be assigned to the service in the case or TCP and UDP services. This can be a single number, the indication ">" (as in >82) meaning whatever port bigger than the number specified, the indication "<" which means whatever port lower than the number specificed (as in <81) and also a range (as in 84-98) which means whatever port in between those 2. The rest of the columns is ignored. For DCE-RPC services, this column should contain the UUID of the service. For RCP services this column indicates the program number. For ICMP Services this specifies the ICMP type. For services of type Other, this column is the protocol number of the service.
Timeout is the timeout for the service (the time after which a session of this service would be considered no longer valid) in seconds. If *default* is specified, then it means that the default timeout specified for all the services of this type in the Global Properties of the SmartCenter, will be applied.

The following is an example of services definition:

```
udp_81,             upd,   81,    default,green,
tcp_bigger_82,      tcp,   >82,   600,
udp_lower_83,       udp,   <81,   default,
tcp_range_84-85,    tcp,   84-85, 1200,   blue,
sample_dcerpc,      dcerpc,12345678-90ab-cdef-0123-4567890abcde,,red
sample_other,       other, 87,    default,red,replies,ip_cmd=RIPCMD_RESPONSE
sample_rpc,         rpc,   100006,    ,   red,
sample_icmp,        icmp,  6,         ,   red,
```

CSV file type and Cluster related objects

Beginning with Object Filler 1.8, cluster objects are also supported with CSV files. To define clusters, there are 2 relevant object types, and the syntax is a bit different: First you've to define the Cluster Members (one line per cluster member), and then you have to define the cluster object itself. However, when defining the cluster object, you have to split it on several lines, indicating instead of the network mask, an object member that belongs to such cluster. All the other parameters have to be the same. The following is an example:

```
clmember1,  member,   10.2.99.1,  255.255.255.255,  blue, , , Cluster Member A
clmember2,  member,   10.2.99.2,  255.255.255.255,  blue, , , Cluster Member B
clmember3,  member,   10.2.99.3,  255.255.255.255,  blue, , , Cluster Member C
clusterA,   cluster,  10.1.2.33,  clmember1,       green, , ,Cluster Object
clusterA,   cluster,  10.1.2.33,  clmember2,       green, , ,Cluster Object
clusterA,   cluster,  10.1.2.33,  clmember3,       green, , ,Cluster Object
```

The first three lines define the cluster members. The last three lines define the cluster itself, and acknowledges clmember1, clmember2 and clmember3 as members of defined cluster "clusterA". Please note that all the fields are the same for the cluster, with the exception of the column to indicate the member.

Please also note than no other Cluster's properties are set (such as synchronization network or cluster topology), so this cannot be used to backup cluster configurations.

CSV file type and Groups

As of Object Filler 1.6, defining simple groups for network objects with CSV files is supported. Since Object Filler 2.0, defining service groups is also supported.

To do this, you must specify the name of the group on the name column (Column 1), the word "group" for network objects groups or "srvgroup" for service groups on the type column (Column 2), and then specify the name of the member on the IP Address column (Column 3). If the member name is not an object that was processed by Object Filler in this file (or predefined), it will reject this member. This behavior is by design, so the user knows that it's trying to include a member that was not created by the file. Following is an example:

```
group1,       group, Int_Srvrs , , , , , ,
group1,       group, Users     , , , , , ,
group1,       group, MyServer  , , , , , ,

tcps1, srvgroup, tcp_81  , , , , , ,
tcps1, srvgroup, tcp_gt90, , , , , ,
tcps1, srvgroup, tcp_lt20, , , , , ,
```

The previous lines will create an Object Group named "group1", whose members will be the previously created objects Int_Srvrs, Users and MyServer. Those lines will create a service group named "tcps1" whose members will be tcp_81, tcp_gt90 and tcp_lt20.

Colors and comments on groups is not supported.

Order is important when you are trying to add groups inside groups. If this is the case, make sure you have created the group you are trying to include as a member inside of another group...

CSV file type and Operations over objects

When using a CSV file as input for Object Filler, some operations over objects are supported. Currently only RENAME and delete operations are supported.

When specifying the RENAME operation, the first object name is the original one, and the last name (the one specified in the Column 3, where usually the IP Address of an object is specified) is thenew one for the object. RENAME doesn't change any object property, such as certificate's FQDN, just the name of the object.

When specifying the DELETE operation, the object name declared on the first column is the one that will be deleted.

The following lines are a sample of operation statements using a CSV files:

```
object1, rename, ObjectA  , , , , , ,
object3, delete,          , , , , , ,
```

The first line will produce the command to rename object1 to ObjectA. The second line will produce the command to delete object3.

Both operations are supported only over network objects. Such operations are not supported by Object Filler on service objects.

CSV file type is the only one that accepts operations over objects.

CSV file type and SmartLSM related objects

Since Object Filler 1.8, defining some SmartLSM related objects is supported. These objects include Dynamic Objects, SmartLSM VPN-1 Edge/Embedded profiles and SmartLSM VPN-1 Edge/Embedded ROBO gateways (types IP40 and VPN-1 Edge X Series). For this to work properly, SmartLSM must be enabled first on the SmartCenter, using "LSMenabler on" command.

The columns here are a bit different, and mean the following:

*** For Dynamic Objects**

- Column 1- name: The name of the dynamic object. Mandatory field.
- Column 2 type: Must be set to "dynamic" (Without the quotes). Mandatory field.
- Column 3 IP Address: IP for this Dynamic object. Please see note below regarding IP addresses for this kind of objects. Mandatory field.
- Column 4 Mask: Must be set to 255.255.255.255 - Mandatory field.
- Column 5 Color: Color for the object. Optional field.

- Columns 6, 7 and 8 are not relevant
- Column 9 Comment: The comment for the object. Optional field.

*** For SmartLSM VPN-1 Edge/Embedded profiles**
- Column 1- name: The name of the profile. Mandatory field.
- Column 2 - type: Must be set to "lprofile" (Without the quotes). Mandatory field
- Column 3- IP Address: IP for this profile object. Please see note below regarding IP addresses for this kind of objects. Mandatory field.
- Column 4- Mask: Must be set to 255.255.255.255. Mandatory field.
- Column 5- Color: Color for the object. Optional field.
- Columns 6, 7 and 8 are not relevant
- Column 9- Comment: The comment for the object. Optional field.

*** For SmartLSM VPN-1 Edge/Embedded ROBO gateways**
- Column 1- name: The name of the ROBO gateway. Mandatory field.
- Column 2- type: Must be set to either "ledge" or "lip40" (without the quotes). ledge means type set to VPN-1 Edge X Series. lip40 means type set to Nokia IP40. Mandatory field.
- Column 3- IP Address: IP for this profile object. Please see note below regarding IP addresses for this kind of objects. Mandatory field.
- Column 4- Profile name: It's the name of a SmartLSM VPN-1 Edge/Embedded profile previously created on this CSV file, or already existing on the SmartCenter. Mandatory field.
- Column 5- Dynamic Object: It's the name of a Dynamic Object previously created or already existing on the SmartCenter. It's mandatory only if you wish to assign an IP or range of IPs to be set as VPN domain behind the created ROBO gateway.
- Column 6- IP or Range of IPs: Only needed and processed if a valid dynamic object has been specified. The IP or range of IPs will be assigned to the dynamic object. If a range needs to be specified, then a dash (hyphen) must be used as a delimiter between the first and the last IP of the given range.
- Column 7- vpn/novpn: If set to "vpn", the previously assigned IPs to the dynamic object, will be exported as part of the VPN topology of this ROBO gateway. Optional field.
- Column 8- Registration key: If specified, this will be set as the Registration Key for this ROBO gateway. Optional field.
- Column 9- Comment: The comments for this ROBO gateway. Optional field.

To illustrate this, the following are some examples:

```
obj_dyn, dynamic ,0.0.0.19,255.255.255.255,blue,          ,     ,     ,Comments
prof1,   lprofile,0.0.0.18,255.255.255.255,blue,          ,     ,     ,Comments
edge_gw ,ledge,   0.0.0.20,prof1,obj_dyn,    192.168.10.40,vpn  ,pass1,Comments
edge_gw2,lip40,   0.0.0.21,prof1,obj_dyn,1.2.3.4-1.2.3.9,  novpn,word2,Comments
```

The first line just creates a dynamic object named "obj_dyn". The IP address specified is necessary. Please see note below regarding IP Addresses. The object is created on color blue and takes "Comments" as the comment for this object.

The second line creates a SmartLSM VPN-1 Edge/Embedded profile. Again, the IP address is necessary. Color for the object is blue and the comments are simply "Comments".

The third line creates a SmartLSM VPN-1 Edge/Embedded ROBO gateway with type of it set to VPN-1 Edge X Series. The IP is needed (please see note below regarding IP addresses). Then uses "prof1" as profile (this SmartLSM VPN-1 Edge/Embedded profile must exist previously on the SmartCenter, or must be created previously on the same CSV file), uses "obj_dyn" as the Dynamic Object, and assigns the IP 192.168.10.40 as the value for this dynamic object. Also specifies that this Dynamic Object belongs to the VPN domain, and sets the Registration Key to "pass1".

The fourth line creates a SmartLSM VPN-1 Edge/Embedded ROBO gateway with type set to Nokia IP40. The profile is "prof1", the Dynamic object is "obj_dyn", but this time the range

assigned to the dynamic object is from 192.168.20.10 to 192.168.20.40 (i.e. a range instead of a single IP). Please note the hyphen (dash) separating both IP addresses. The Registration Key is set to "word2".

Important note regarding IP Addresses for SmartLSM related objects
When building SmartLSM ROBO gateways with the command line, specified IP Addresses for the objects must be in the range from 0.0.0.10 to 0.0.254.254. Please note that you *must make sure* that no duplicate IPaddress exist on the configuration. To assure this, please log in to the SmartLSM GUI, sort the elements by "ID" (second column from left to right) and make sure the IPs you are specifying are not listed there. Then, use Object Dumper to dump the contents of your current Objects_5_0.C file and see that no profile or dynamic object is already using the IPs you're trying to assign.

To avoid any problems or IP conflicts in any case, is **highly** recommended to have the SmartCenter Server clean of SmarLSM objects, dynamic objects, profiles or any kind of dynamic objects. This is, you should use Object Filler just for the initial configuration, unless you know what you're doing.

CSV file type and importing security rules

Since Object Filler 1.9.2, importing basic security rules from CSV files is possible. When importing security rules, the syntax for the line is the following:

```
security_rule,source,destination,vpn,service,action,track,install_on,time,
comment
```

Where:
security_rule is a key word that specified object filler to treat this line as a security rule definition. It must be like this.
source is a network object. You can specify several, using a semicolon (;) as separator. Currently, having user groups as source (for authentication rules or VPN rules) is NOT supported.
destination is a network object. You can specify several, using a semicolon (;) as separator.
vpn is the VPN communities for this rule
service is the service object for this rule. You can specify several, using a semicolon (;) as separator. Currently using resources (uri, smtp, etc.) is NOT supported.
Action can be accept, log or drop. Any other action is NOT supported
Track can be Log or None. Any other action is NOT supported.
Install_on can be any Check Point gateway object, or the word "Any". You can specify several, using a semicolon (;) as separator.
Time can be a time object, or the word "Any". You can specify several, using a semicolon (;) as separator.

The following are examples of valid rules defined:

```
security_rule, Server1,      Srv2,      Any, tcp_81,     accept, log, Any, Any,
security_rule, Host_X;HostY, Any,       Any, http,       accept, log, Any, Any,    XYZ
security_rule, Internal_LAN, Srv1;Srv2, Any, NBT,        accept, log, Any, Any,    Comment
security_rule, LocalMachine, Any,       Any, icmp-proto, drop,   None,Any, Any,
security_rule, InternalNet,  Any,       Any, ftp;telnet, accept, Log, Any, Any,
security_rule, Any,          Any,       Any, Any,        drop,   log, Any, Any,
```

The processing of rules is affected by the Object Filler switch –nopv – If this switch is not specified, Object Filler will try to check that the objects specified in the rules were processed before (or are part of the predefined objects). If they were not processed (or predefined), they will be substituted by "Any".

If –nopv is specified, the checks mentioned above are not performed.

Section headers are properly recognized and processed both by Object Dumper, and Object Filler while importing rules from CSV Files. If a section header is specified, it should be done using the keyword *section_header* in the first column, instead of *security_rule*, as in the following example:

```
section_header, OPSEC_rulebase
```

List (list) file type

Specified file must contain 2 mandatory fields: IP Address and netmask. Additional optional columns are color, IP behind which NAT will be done, object behind which NAT will be done, and NAT type (Hide or Static). The explanation for all those columns is exactly the same as for the CSV file type.

Object Filler automatically calculates (based on provided netmask) if the object is a network, a host, an IP Address range or a group, then generates a name (unless it's a Group, for which the name it's expected to be the first parameter) and the appropriate network object. Due this, the only supported object types are hosts, networks, IP address ranges, and groups. Check Point Hosts, Check Point Gateways, Check Point Dynamic Gateways, Plain Gateways, Interoperable Devices, OSE devices and the others are not supported on this type of file. If you need those, please take a look on the CSV file type.

When building IP Address Ranges, you must enter the starting range IP on the first (IP Address column) and the ending range IP on the second (netmask) column. Ending IP must be "greater" network-wise than the starting IP.

If you whish to build Groups, all you have to do is to specify the name of the member on the IP Address column, and the IP of the member on the netmask column. If the member (i.e. if the object corresponding to this IP) was not processed before in this file, Object Filler will reject this member. This behavior is by design, so the user knows that it's trying to include a member that was not created by the file. You cannot include groups as members of another group while you are importing a List type of file: this is supported only with CSV files. Also, comments and color for groups is not supported.

Hosts (hosts) file type

Indicated file has the format of a hosts file (/etc/hosts on Un*x systems, or %SYSTEMROOT%\system32\drivers\etc\hosts on Microsoft Windows systems). Object Filler automatically generates hosts objects using the name and IP listed on the file.

When importing hosts files, you may specify an object type besides plain hosts in the Object Filler's command line with the –t switch, so you can actually build OSE Devices or Plain Gateways for example.

Cisco PIX (pix) file type

When importing from Cisco PIX, following versions were tested: 5.1(1), 5.1(2), 5.1(4), 6.1(4), 6.2(2), 6.3(1).

The file entered as input for this option is the configuration listing from a Cisco PIX device. You can get this information from a PIX device using the command "show running" or "write terminal".

When importing from Cisco PIX, Object Filler will only recognize plain Hosts, OSE devices (for the interfaces of the PIX device itself) and Networks. Object Filler will recognize all valid IPs that are listed in the configuration, not only those from the rulebase, and will process it.

Names on the objects are assigned according to the object type recognized (OSE, Host or Network).

By default NATted objects (Static or Hide) are supported. As a matter of fact, Object Filler by default processes the NAT statements first

In the case of static NAT, NATting to the outside interface it's privileged, this is, if the same IP is NATted on several interfaces, Object Filler will try to leave as the imported NAT the one that faces to the outside interface. If no outside interface is declared, then the first static statement found is applied.

If several global statements are bound to the same NAT ID, only the first IP of all of them will be used, and the outside interface will be preferred also.

In the other hand, all NAT statements are processed. If several NAT statements belong to the same NAT ID, all of them are processed to the first global IP specified for such NAT ID, as explained before.

Object Filler won't process ranges (i.e. when IP addresses are in the format aaa.bbb.ccc.ddd-www.xxx.yyy.zzz - Example: 1.2.3.4-1.2.3.10). In those cases, the program will split the range and will take in account just the first IP (1.2.3.4 from our example).

NAT processing is also affected by the –nonat switch of Object Filler. If this switch is specified, no NAT processing will occur at all.

Since Object Filler 1.9.2 the import of rules from Cisco PIX configuration files is also supported. The only supported rules that may be imported are the ones specified with the *access-list* statement. To make this happen, you have to specify the –p (policy) switch in the Object Filler's command line.

If there are several access-lists in the same configuration, all the access-lists will be imported, but they will be separated using a standard policy tag in the imported configuration

To open the imported policy in the SmartDashboard (once you have imported the configuration via dbedit), go to File, Open. You will see the Object Filler imported policy there.

Juniper/NetScreen ScreenOS (netscreen)

When importing from NetScreen devices, ScreenOS from NS5XT, NS100, NS500 and NS5200 devices were used for testing. Tests have been conducted using ScreenOS 4.X and 5.X versions of the OS.

The file entered as input for this option should be the configuration listing from the NetScreen device. You can get this information from a NetScreen device using the command "get config all".

When importing from NetScreen, Object Filler will only recognize Hosts and Networks as valid types. Object Filler will recognize all valid IPs (not only those from the rulebase, but any IP) and process it. However, only Check Point gateways (for the IPs of the device itself), plain Hosts and Networks will be recognized and built.

Names on the objects are assigned according to the object type recognized (Check Point Gateway, Host or Network)

Static NATted (mip) objects are supported. Hide NATted (dip) and PAT (vip) objects are not supported.

SecureComputing Gauntlet (gauntlet)

Importing from Gauntlet was tested with version 5.5 running over Solaris. The configuration files needed may be found under /usr/local/etc/mgmt - but this may change.

Newer Gauntlet versions should work, but were not tested. Any reports of Object Filler running over other versions would be appreciated.

Only hosts and subnets are recognized, no other types of objects are built. Name is not imported from file. Instead, a new name is built according to the object type recognized.

No NAT conversions are done while converting from Gauntlet, mainly because of the lack of more testing files.

If you need to convert from Gauntlet and have some problems, have sample files willing to share, or have documentation of something unsupported on Gauntlet that should be here (like Groups, NAT support), please send me an e-mail.

SecureComputing SideWinder (sidewinder)

When importing from SideWinder, version 5.21 patch 9 configuration files were used for testings, with the contents of both ACL and IPFilter settings.

When reading the ACL configuration the following tables are supported: ipaddresses, hosts, subnets, and netgroups. Domains and servicegroups are not supported yet.

When the object has a name (for hosts, subnets and netgroups), this name is kept on the build object. If object has not a name, a name is created according to the object type recognized.

Hosts, networks and groups are properly recognized.

Object Filler also takes the IPs found on ACL or IPFilter statements. When importing the IPFilter statements, NATted IPs are converted properly, always using Hide NAT.

Symantec Raptor (raptor)

When importing from Raptor, version 6.03 for Windows was tested.

The file used is the gateway.cf, which contains the IPs and rules used for the configuration.

Hosts and networks properly recognized, as well as declared TCP and UDP services that are declared by port and have no name on them.

No NAT statements are supported on this version.

Cisco IOS Router (ciscorouter)

When importing from IOS configurations, versions 11.0, 11.2, 11.3.3.T, 12.0, 12.1 and 12.2 were tested.

Hosts and networks are properly recognized. No NAT statements are supported on this version.

If –p switch is used in Object Filler, and the configuration contains rules, the rulebases are processed accordingly.

Importing Object Filler's output to a Check Point SmartCenter Server or Provider-1 MDS Server

Modifying Object Filler's Output before importing

Since all output is directed to a text file, it's feasible to edit this file using any text editor, and modify (as an example) the prefix for the object's names (Net for Network, or Host for IP; as examples) or do any other modification you may need. This is true in both cases: for CSV formatted output (-a switch) and for dbedit commands output (-o switch).

Using dbedit to process Object Filler's results

First of all, it is greatly suggested you to read the following articles on the public partition of SecureKnowledge:

- https://secureknowledge.checkpoint.com/sk/public/idsearch.jsp?id=skI3301
  Editing the object_5_0.C file using the dbedit utility

- https://secureknowledge.checkpoint.com/sk/public/idsearch.jsp?id=sk10104
  Using the dbedit utility to modify the value of a specific network object property

- https://secureknowledge.checkpoint.com/sk/public/idsearch.jsp?id=sk12222
  Using queryDB_util to query the database

If you have access to the registered partition of SecureKnowledge, you may find the following articles also useful and interesting:

- Using dbedit utility to create network, host and group objects, and place network and host objects in group objects
  Solution ID: sk22957

- Creating Service Groups, Services, and Adding Services to Groups using DbEdit
  Solution ID: sk30370

- Using the dbedit utility to modify the value of a specific network object property
  Solution ID: sk10104

- Running command line 'dbedit' in a CMA environment
  Solution ID: sk23802

- Update command fails to execute properly using the dbedit utility
  Solution ID: sk10098

- Downloading and installing Check Point Database Tool utility
  Solution ID: sk13009

Then, it's important to remember that Object Filler's output files can be transferred from one machine to another. So it is **not** necessary to have Object Filler running on the same machine where the target SmartCenter Server is sitting. This SmartCenter Server could be in a different machine, and even a different operating system than the one used to run Object Filler.

If you are going to use the Object Filler's dbedit commands file result in a different machine from the one used to generate it, please verify that the proper dos2unix conversions (converting CR+LF to CR only) have been done, when you are passing files between machines with different operating systems (Windows to UN*X).

Keep in mind that dbedit commands are 100% ASCII text and should be treated accordingly when transferring using FTP-like mechanisms. If you start to see an error "Token contain illegal character" then you're probably transferring the file in the wrong format. Please verify that, if you're using FTP, you establish the transfer mode to "ASCII" instead of "binary" (which is the default sometimes). If you are transferring the files using diskettes, and the source and destination machines have different operating systems, dos2unix conversions may also apply.

Besides that, when importing files to the SmartCenter using dbedit, please make sure that:
- Your management processes are up and running. In the SmartCenter Server machine you can use the command "`cpstat mg`" or "`cpstat mg –h <IP address>`" to verify it.
- Your SMART Client (GUI clients), especially the SmartDashboard, are not running. If you strictly need to use them while importing, then please log in to the SmartCenter Server as read-only while you do the import.
- You are using a user with administrative privileges at operating system level (root, admin, Administrator or equivalent) If not, then change to a higher privileges user or a user that has enough permissions to run Check Point's binaries and affect Check Point's configuration.
- The IP from which you are running dbedit is declared as a valid Smart Client (GUI Client) IP. If not, then add it using cpconfig. In Provider-1 environments, you may need to also add the IP addresses for the MDS and/or the CMA itself as GUI Clients into the target CMA's configuration.

If you are running Provider-1, besides the above, also make sure that:
- You're doing the process on a MDS Manager or MDS Manager and Container server.
- You set the proper environment (using "mdsenv cma") before trying to connect using dbedit.
- You use the CMA's IP address as target for dbedit (dbedit's switch -s), and NOT the MDS IP Address.

You may try to run dbedit first and see if you can get into the target SmartCenter/CMA without any problems. Then you simply have to import the file using "-f" switch from the operating system command line, like in the following examples:

```
dbedit -f output_sample.txt
```

In the above case dbedit will read input from the file "output_sample.txt". This will prompt for the SmartCenter Server IP Address, an administrator username and the administrator password.

```
dbedit -s localhost -u admin -p duckystyle -f nat_networks.txt
```

In the above case dbedit will read input from the file *nat_networks.txt*, specifying that the SmartCenter Server is the located at the localhost, using *admin* as administrator's username and *duckystyle* as admin's password.

```
 dbedit -s 10.20.30.55 -u ccse -f nat_networks.txt
```

In this case dbedit will read input from the file *nat_networks.txt*, specifying that the SmartCenter Server or CMA is located at the machine with the IP 10.20.30.55 using *ccse* as administrator's username and asking interactively for the administrator's password.

If you get any error message or weird behavior while trying to import the objects you created with Object Filler, please consult Appendix A to see common causes of known problems.

Object Dumper

Program syntax:

```
odumper help  (prints help pages)
odumper -f file [-p file] -o file  [-d] [-html] [-v]
```

-f specifies the path to the objects (Objects_5_0.C or objects.C) file you want to process
-p specifies the path to the rulebases (rulebases_5_0.fws) file you want to process - Optional
-o specified the path to the output formatted file you want to have
-d tells the program to also print the default objects - Optional
-html formats the output to HTML (instead of default CSV format) - Optional
file is a valid filename - such as output.txt, output.html or objects.C

Required parameters: -f and -o

If you want to redirect the program's output, you can use the operating system ">" operand to do so.

Please note all parameters are case sensitive.

Program syntax #1 : Asking for help

```
odumper help
```

Prints every possible command line combination.

Program syntax #2 : Importing from an Objects_5_0.C, rulebases_5_0.fws and/or objects.C file

```
odumper -f file [-p file] -o file [-d] [-html] [-v]
```

Please note all parameters are case sensitive.

-f - input File - It can be an Objects_5_0.C file taken from the $FWDIR/conf (or %FWDIR%\conf) directory from a SmartCenter Server. It can also be an objects.C file taken from the $FWDIR/database (or %FWDIR%\database) from a Check Point Gateway (Enforcement Point) in a distributed configuration. Also you may use Check Point FireWall-1 4.1 objects.c files (located under $FWDIR/conf/objects.C) From this file the program reads the objects definitions, so they can be displayed after. **Required parameter**

Example:

```
odumper -f copy_of_Objects_5_0.C -o output.csv
```

-p - Policy File - It must be the rulebases_5_0.fws file, taken from the $FWDIR/conf (or %FWDIR%\conf) directory from a SmartCenter Server. From this file the program reads the rules definitions. **Optional parameter**.

Example:

```
odumper -f copy_of_Objects_5_0.C -p Copy_of_rulebases_5_0.fws -o output.csv
```

-o - Output file - The name of the file where resulting objects information will be stored. Please make sure you have enough disk space to store all produced information. To calculate this space, take an average of 150 bytes per object to process. **Required parameter**.

Example:

```
odumper -f copy_of_Objects_5_0.C -o output.csv
```

-html - HTML format for the output file - when this switch is specified, the output written to the file specified by the -o switch, is formatted on HTML using tables, and can be viewed by any standard web browser. Mozilla 1.7, Internet Explorer 6.0 and Netscape 7.2 for Windows were tested. **Optional parameter**.

Example:
```
odumper -f copy_of_Objects_5_0.C -o output.htm –html
```

-v - Verbose mode - shows in the console (the standard output, the screen) details on how the processing is being done line-by-line. This is very useful especially when debugging, but not in other circumstances since the output can be really overwhelming and a bit meaningless for most of the times. **Optional parameter**.

Example:
```
odumper -f copy_of_Objects_5_0.C -o output.csv –v
```

Modifying Object Dumper's Output and Importing Back

Since all output is directed to a text file, it's feasible to edit this file using any text editor, and modify anything there. However, due the format used (Comma Separated - CSV), it's more easy to edit files produced by Object Dumper using any spreadsheet program able to open CSV files, such as Microsoft Excel.

Files produced with Object Dumper, can be converted to dbedit files again, using Object Filler's CSV option (-i csv). Any modifications made to the file can be imported back to the SmartCenter this way.

Remember, that if you are modifying a configuration to import it back with Object Filler, you should change the object types accordingly: *modhost* instead of *host*, *modnet* instead of *net*, and so on. Please see the table of supported objects for modifications in the beginning of this document.

Web interface for Object Filler and Object Dumper

Both programs have a single, not fully featured and shared *proof of concept* web interface, which is provided here in two files:
- ofiller.html - Is the HTML code that acts as front-end for the user.
- ofiller.pl - Perl Code that processes as CGI module, all the data captured by the front-end.

To make this web interface usable, you must have Perl installed on your computer. In our case we tested using the Perl package provided by ActiveState found here: http://www.activestate.com/Products/ActivePerl/ when testing on Windows, and the Perl distribution provided with Red Hat Linux 7.2 while working on GNU/Linux. You need also to have a Webserver running. This was tested using Apache Web server 1.3.27 for Windows and for Linux, and also Internet Information Server (obviously under Windows).

You should place ofiller.html inside a public HTML folder, available for document publication from the webserver you are using. You should place ofiller.pl on the cgi-bin directory for such webserver.

It's also necessary to modify the following lines inside ofiller.pl:

* my $PATH_TO_EXEC
  This should reflect the path where Object Filler and Object Dumper executables are available (ofiller.exe and odumper.exe, or their GNU/Linux versions). It should not include the program names themselves, just the path.
  Examples:

```
my $PATH_TO_EXEC = "d:\\ofiller\\cgi-bin\\";
my $PATH_TO_EXEC = "/usr/local/ofiller/";
```

* my $OFILLER_EXE
  This should contain the name of the Object Filler executable. Examples:

```
my $OFILLER_EXE = "ofiller.exe";
my $OFILLER_EXE = "ofiller.lin";
```

* my $ODUMPER_EXE
  This should contain the name of the Object Dumper executable. Examples:

```
my $ODUMPER_EXE = "odumper.exe";
my $ODUMPER_EXE = "odumper.lin";
```

* my $UPLOAD_DIR
  This should point to an empty directory that must exist before the program can be run. It is used to upload and process configuration and Objects_5_0.C files that will be processed by those tools. Examples:

```
my $UPLOAD_DIR = "d:\\ofiller\\upload\\";
my $UPLOAD_DIR = "/usr/local/ofiller/upload/";
```

I would like to greatly thank Pedro Paixão, Check Point Latin America Regional Technical Consultant & SE Manager for his help on this web interface.

Appendix A. Frequently Asked Questions

Here is my try to condense questions that I get asked most of the time, in an attempt to provide fast and concise answers. Please feel free to write me an e-mail if you don't find an answer to yours. I promise I'll reply to it.

1. General Questions

Where can I get the latest versions?
Please consult the section "Programs Availabilty" above on this document.

Who maintains Object Filler and Object Dumper?
So far, Martín Hoz (mhoz@mexico.com)- Security Engineer for Northern Latinamerica at Check Point Software Technologies, is the person that maintains this (but this may change in the future), not without the valuable help of persons that assisted a lot with information on the Check Point products and also people that tested it version after version. "thanks".

Are these tools supported by Check Point or some other entity?
No. These tools are not officially supported by Check Point nor somebody else, in any way. Please use them at your own risk. Any good or bad result of using these tools either directly or directly is only responsibility of the person using them. Please read the Disclaimer included in this file for more information...

Got a problem, can I e-mail you?
Sure! - and I always answer my e-mails. But please, before doing so, read the provided documentation and make sure you're using the latest version of the tool, as I regularly audit and fix the code (i.e. try to make it support "can't happen" situations) while adding new features. The list of available sites to download the latest version of the tools is listed above.

What is the origin of Object Filler?
Pain on my fingers. Really. As a presales systems engineer, several times found myself on the duty of filling the SmartCenter Server with tons of objects which was a bit tiring, boring and painful doing it by clicking all the time. So I thought of more automatic way on doing it, and also trying to ease the process of importing configurations from other brands (something also needed every now and then - and more and more frequently in recent times, once people realize why Check Point is superior ;-). Given the open and robust nature of SmartCenter and the powerfulness of dbedit, this task was not hard at all... and seemed natural to be done…

What is the origin of Object Dumper?
Just to have another tool to dump the Objects_5_0.C and rulebases_5_0.fws on a more readable and easy to manipulate format, and also have a companion tool for Object Filler for exporting and importing back configurations.
Object Dumper is not intended to be a tool to document the configuration of your SmartCenter. If what you are looking for is a documentation tool, I strongly recommend you to take a look on a couple of good tools that do similar (even better in my opinion) job on this:

- Web Visualization Tool:
  http://www.checkpoint.com/techsupport/downloadsng/utilities.html#visualization
  Officially supported by Check Point, that supports exporting rules and objects to HTML/XML format.
- FW1Rules:
  http://www.wyae.de/software/fw1rules/
  Unsupported by Check Point tool, that allows to export both objects and rules in several formats, including HTML and CSV. Written on PERL.
- CPrules:
  http://www.wormnet.nl/cprules/
  Unsupported by Check Point tool that allows to export the rulebase to HTML. Written on PERL.

Object Dumper was not intended to be a backup/restore or migration tool either. Please check the Related Programs section in this document, as well as the Object Dumper section below, to find out more on this.

Why "Object Filler"? Why "Object Dumper"? - Where did you get those names?
Just names that came to my mind. I thought they were original and had a meaning for what the tools intend to do. Sorry if they look ugly to you. :-P

Are there any minimum requirements (processor, RAM, disk, etc.) to run these programs?
Not really. The program uses the disk to store the output, so just make sure you have enough disk to store this. Having 1 GB of free disk would be way more than enough and safe in most of the cases. In the other hand, the program itself plus the internal data would need around 32 MB on RAM (assuming you will process *thousands of objects*). Generally speaking, having these resources will be more than OK. One last word on memory though: On PCs with less than 128 MB of RAM I've been informed that sometimes the program just doesn't do nothing and stops with no error message or anything. I'm still trying to catch up where the bug is, so if this is your case, please let me know. Processor will just speed up calculations, but since both programs are real small and not intensive on processor usage, any processor is ok. Same goes to the bus.

Why did you prefer to parse text files and to use dbedit, instead of using the CPMI OPSEC API?
Because of three reasons: I like it better the way it works now, and it simply works (remember I'm doing this for fun! ;-). Then because it's clearer to people how it works and what the results are (so, it's easier to make people to trust in the tools, because they understand what is done and how is done), and allows them to change things if they want to; and finally because to me this was always and (still is) just a proof of concept, which is always easier to do with text. I never thought Object Filler & Object Dumper would be at the stage they're now... and I honestly never thought somebody else would use this except me and a couple of friends, so it was more like a personal toy & hobby, but there you go...

What development environment/programming language are you using to develop these?
I use standard ANSI C for programming. Why not PERL or something else? Because in general I don't like interpreted languages like PERL (except SHELL scripting). Also, because I don't know (and unfortunately don't have the time to learn) PERL. Then, because at this time I only remembered C programming, because I like C, and finally because I want it to be C. Right? ;-)

Generally speaking, I don't really use a development environment. While on Microsoft Windows, I use gvim (http://www.vim.org) and GCC 3.4.3 (I use the DJGPP flavor - http://www.delorie.com/djgpp/) – occasionally I also use the Visual C++ Toolkit 2003 (which includes a free command line compiler). While doing stuff on Solaris and GNU/Linux, I use an standard GNU vi and GCC 3.2.2

Are these tools available for free (at no cost), or do I need to pay something?
Yes, they are free if what you're asking is if they are available at no cost. They are not really free in the sense of freedom, since they really belong to Check Point Software Technologies as intellectual property (even if they are not officially supported) because they were written by a Check Point employee (me). However, if Object Filler or Object Dumper were useful for you, I would like to hear about it, so I'd ask you to "pay back" giving me feedback about your experience (especially in environments not documented as tested on)...

I really want to payback somehow with money or something like it... How can I do it?
I extensively use gvim (http://www.vim.org), a free text editor which asks in return to support poor children in Uganda through buying a book or making a donation (I already bought the book). I'd ask you to participate if you want to payback with money, either buying the book (if you like vi you will love to use gvim, which I recommend, and the book gives you interesting ideas, so it won't be a waste of money, and you will be helping) or donating something. Here is the link: http://iccf-holland.org/click5.html - In México we also have poor people (unfortunately, like everywhere, but here sometimes is also really bad), so you can also donate to the following organizations that somehow assist poor child and people in need in general, in México and other places also. :
http://www.mexico-child-link.org/
http://www.cruzrojamexicana.org/donativos/portarjeta.php
http://www.redcross.ca/
http://www.redcross.org/
http://www.unicef.org/
http://www.oxfam.org/
http://www.savethechildren.org/
http://www.msf.ca/

Remember that there's always somebody in this world that needs our help. You needed the assistance of this tool, and some people needs our money to have some food or better means on this life: I trust the tools saved you enough bucks or time to make you willing to give a (even small) donation to any of those organizations (or even better, all of them ;-) or in general, to any other organization that tries to make this world better somehow…

Finally, if you really want to send me a gift, I enjoy very much getting postcards from everywhere in the world, so if you have the time, please send me one... (but please, only if you already donated something to the charities above listed ;-) - As of February 2005, my current mail address is:

Martín Humberto Hoz Salvador
Querétaro 162 Depto. M-203 esq. San Bernabé
Col. Progreso Tizapán.
Del. Álvaro Obregón
CP 01080
Ciudad de México, Distrito Federal – MEXICO

Do you have the source code of the tools available for users to read it or modify it?
No. I can't distribute the source since it doesn't really belong to me, as I explained in a question above. Sorry.

How big is the source code for both tools?
So far, this is the wc (word count) report for the source code of the tools:
For Object Filler
20,110 lines; 62,061 words; 732,114 chars
For Object Dumper
5,147 lines; 14,289 words; 176,133 chars

Can I redistribute these utilities on my website/ftp site? Can I redistribute these utilities together with my package or software?
If you will charge for that in any way or lock the distribution in any way, the answer is definitely no. Otherwise the answer is a maybe. Send me an e-mail (address available at the Contacting the Author section) if you're planning and have the way to do so and we'll discuss it.

On what platforms can these utilities run?
Currently the platforms on which the tools run natively are Microsoft Windows, GNU/Linux (Red Hat Linux) and Check Point SecurePlatform. I was informed that it also runs on other Linux distributions like Mandrake or Suse, but not confirmed it by myself. I'm planning to add support for Solaris for next versions if there's some demand. However, remember that you can put and user the output of them wherever you want. I've done it with Nokia IPSO and Solaris itself for example.

On what Check Point versions were these tools tested while under development?
2.0 version of the tools was tested on Next Generation with Application Intelligence R55; mainly. Also was tested a bit on NG+AR R55W.

2. Object Filler

What's the best way to invoke Object Filler when importing files?
Use always the "-v" option, and then send the output to a file. Then review with a text editor such output file to see the details of what happened, and look for possible errors on the processing. The syntax should be something like this:

```
ofiller -f import.csv -i csv -o objects.dbedit -v > output.txt
```

Then edit it:
`notepad output.txt` (or "`vi output.txt`", or "`edit output.txt`", or whatever is needed according to your text editing preferences).
This way you will be able to see how the processing was done line-by-line.

I also strongly recommend exporting your current list of objects, and comparing it to the one that will be imported. This comparison may be accomplished using the ASCII output mode (-a) of Object Filler, and using Object Dumper to export your current configuration to CSV. This way you will notice which objects have chances to be modified, *before* you do any changes to your live configuration.

A good tool to compare text files that I like is CSDiff, which you can find here:
http://www.componentsoftware.com/products/csdiff/index.htm

Why building SmartLSM VPN-1 Edge/Embedded ROBO gateways or profiles, doesn't work in my SmartCenter?
Please make sure you have SmartLSM enabled on your SmartCenter. To do so, use `LSMenabler on` command from the Operating System command line, on the machine where your SmartCenter sits. Also, please keep in mind that this feature was developed and tested for NG+AI R55. If you're trying with a newer version and it doesn't work, please send me an e-mail.

Is it possible to modify the IPs of a massive number of objects (i.e. change all 192.168.x.x objects to 172.16.x.x) somehow using Object Filler?
Yes, in combination with Object Dumper. First, export your objects information with Object Dumper. Then, using any Spreadsheet or text editor program, edit the file and select the objects you are interested on changing. After that, using the search & replace facility of your editing program, change the IPs you want to change (in the example case, use "search '192.168.' and replace with '172.16.'"). Also replace the type of the object, with a preceeding "mod". For example, if the object's type is "host", replace it by "modhost". If the type is "net", replace it by "modnet" and so on. Then, import this information again using Object Filler (and CSV file option). This will do the trick. Don't worry about other property of the objects (like certificates), since the only property of the object that will be modified is the IP Address. If you would like to modify also other properties (Certificates, etc.) Object Filler can't help you on that.

Why Object Filler doesn't support importing rules from NetScreen, Gauntlet, SideWinder, etc.?
In previous versions, no rule importing was possible at all. This was mainly because of two big reasons:

- Because when migrating is a good chance to review the rulebase, so it's better to review what you're going to configure in your brand-new Check Point VPN-1 Pro/Express.
- Because doing rule translation between firewalls is not easy ;-) especially when the philosophy is different (example: proxy or packet filter, versus Stateful Inspection)

Since version 1.9.2 it supports importing basic rules from Cisco PIX and Cisco Routers, as well from CSV Files.

If you think Object Filler should support importing rules from other brands, please send me an e-mail (explaining your reasons too), and if I get enough requests, I'll try to do it! :-) – if you do so, please send me also example files of the brand you wish to support, as if I get more information on the file structure, my job will be easier and you will get results faster ;-). You may sanitize such file changing IP addresses or names, just please keep the file structure, so I can work on it.

Are you going to support importing configuration from X brand of firewall, or X type of file?
May be. When I released Object Filler1.2 I thought that it would be the last release ever. Then I got myself some other tasks that could be eased using Object Filler and then decided to increase the functionality, including new types of objects and other configuration files for other brands. So, if you have suggestions on what other file types Object Filler should support and/or you have sample configuration files for other firewall brands (you may sanitize them by changing names and IP addresses if you want, but if you do so please keep the file structure), or simply something you think can be eased with some extra functionality on the tools, please send me an e-mail.

Can I help on the process of supporting a new file type?
For sure! - if you have samples of the type of file you would like to support, you can submit the files to me. Please send me an e-mail with this information. I don't need the real names or IP address information (so you can use search & replace of notepad, vi or whatever, to substitute that and "protect the innocents"). Just please leave the format of the file intact, so I can analyze it correctly and find the proper pattern matching for it. And of course, if you allow me to, I'll mention your name on the thanks section. ;-)

What happens if while importing a configuration file, I choose the wrong type? (i.e. if I choose Cisco PIX when in fact it's a SideWinder configuration).
Object Filler will process it, but just not the right way. Usually NATs statements, group associations (when available), proper netmasks, and other information won't be processed the right way. In all cases, Object Filler always tries to figure out if the current file is of the right type. If the program detects that it's not, then will tell about this suspicious status, but this mistake detection mechanism is not 100% reliable, so always try to use the proper option.;-)

Why is the different the number of imported objects reported by Object Filler than the exported ones reported by Object Dumper? I compared them while running over the CSV file that Object Dumper just exported from my SmartCenter…
Most of the times, duplicates. Object Dumper doesn't apply too much verification while exporting objects, but Object Filler does while importing them. So, if you have the same IP address under different names, or the same port number under different names for example, Object Filler will process just the first one found and will complain and report the others as duplicates/invalids. I know these kind of duplicates are something totally permissible by SmartCenter and valid from the operations point of view, however, I just wanted to make sure that people knows (once more) they have duplicates, while importing. ;-)

How do I create a CSV file with Microsoft Excel to import it later with Object Filler?
Just create a new spreadsheet, and follow the column order described previously on this document.

Instead of saving it as a usual spreadsheet, select "Save As" and then choose the CSV Format (usually represented as "Comma Separated Values", "CSV File" or something like it). If you do this, please just make sure that the resulting file doesn't have quote signs (") also as field separators. If it has quote signs, then just remove them. If you don't remove such quote signs, it will result in problematic behavior of the tool.

Are you going to support *X* type of object?
May be. Depends on feedback... – If the tools currently don't support an object type you need, please let me know...

Why the tools don't simply support all the known type of objects for once?
For 2 basic reasons: I want to keep the code as simple as possible. Keeping support for all (even rarely used) known objects, adds complexity to the code (making it harder to maintain) and to the usability of the tools. That's why I rely on your feedback to do something extra. In the other hand, some objects may change (definition, naming convention, properties) while the most used ones are less likely to change. Having fewer changes in the code leads (again) to more stable tools, and more usability on them.

Why Object Filler doesn't support users?
Because there's already a way to do so, it's officially supported and it's well documented: Use "fw dbimport" and "fw dbexport" for that. The SmartCenter and the Command Line documentation have good information on this.

3. Object Dumper

So, What's the main purpose of Object Dumper? Doing backups or migrations?
No. Object Dumper was created more to assist *a bit* on documentation stages, but mainly to make day-to-day operations easier in conjunction with Object Filler, especially on bulk imports, modifications or transports. When Object Filler needs some information from the current configuration to do a job, Object Dumper is supposed to provide this. Definitely I did not have in mind backups nor complete migrations purposes, even though I've got several reports of people using it for real world migrations on relatively simple (even though some of them real big, with hundreds of thousands of objects) installations.

Why Object Dumper is not good for doing backups, policy merges, upgrades or migrations?
First, because it's not supported and you want to have Check Point support backing you if something goes wrong. But also, because Object Dumper won't give you important information you would need in a restore case. For example: Object Dumper won't export important object properties such as certificates or particular VPN settings. For backups and migrations the cpmerge, upgrade_export and upgrade_import tools available at Check Point's web site, are by far much better: more powerful, more easy to use, more focused precisely on that, and especially they are officially supported. You can find those tools and their respective documentation here:
http://www.checkpoint.com/techsupport/downloadsng/utilities.html

Does it work with 4.1 objects.C files?
Yes, it does work. I've done some testing with VPN-1/FireWall-1 4.1 objects.C files (located under $FWDIR/conf/objects.C) and it works recognizing hosts, networks and some services, but it has not been fully tested. This is only for objects of types hosts and networks , TCP and UDP services. Rulebases file from 4.1 Check Point products has not been tested at all.

Why Object Dumper doesn't support users?
Because there's already a way to do so without it, it's supported and it's well documented: Use "fw dbexport" and "fw dbimport" for that. Please refer to the Check Point SmartCenter and Command Line documentation for more information on such commands.

4. Common problems

Why I can't see CSV files on columns when I open them with Microsoft Excel?
Try this: Go to "Data" Menu, choose "Get External Data" or "Import External Data", then "Import Text File" or "Import Data". Please select the file you're trying to import and press "Import or

"Open". If you're on Microsoft Office Excel 2000, then choose "Delimited", then press "Next". Now choose "comma" and then press "Finish". If any additional windows appear, just press "OK". That should do the trick.

I'm getting the error *"'@'network_objects' - Token contain illegal character - Invalid Object Name* while trying to import the dbedit file produced by Object Filler. I check the file and it seems to be ok. What is going on?
Almost for sure you're transferring the file to another machine in a different platform/operating system. Please make sure that while you're doing so, you're transferring the file as ASCII TEXT. If you're doing the transfer via FTP, remember that some clients/servers behave with Binary (bin) transfer mode as default. Change the transfer mode to TEXT (ASCII) before transferring the file. Using the TEXT (ASCII) mode to transfer the file will fix the problem, if this is the cause. If it is not your case, please let me know.

I'm getting an error while importing the dbedit file, that says *Error... syntax error in line NNN Aborting.* - I look in the file for such NNN line number, and it's a blank line. What can be wrong?
The blank line itself is wrong. The dbedit utility will always complain if a blank line is found in the file used to specify commands to be executed. Usually this kind of thing happens when you copy-paste the contents of the dbedit commands file that results from the Object Filler execution. Normally Object Filler doesn't append this blank line. If you add it by accident while copying-pasting, you may safely ignore this message.

What does error *network_objects::XXXXX Object XXXXX already exists* or *Object XXXXX already exists* means? – I get it eventually while I'm importing the objects
This error means that dbedit got the instruction to create an object that already exists. This error is very frequent when you export the current configuration via Object Dumper, and then try to import back modifications in the objects via Object Filler, but forget to change the object type to a *mod* object (i.e. use *modhost* instead of *host*, *modnet* instead of *net*, etc.)

While I'm importing a big dbedit file I get several errors in a row. The messages say something like *A disk error occurred during a write operation*, *Failed To Send Audit Log*, *Failed To Send Audit Log for network_objects:: XXXXX* or *network_objects::XXXXX Object XXXXX already exists* – I also notice that not all the objects/rules that are supposed to be processed, were taken correctly. I have a large (hundreds, thousands) number of lines in the dbedit commands file being used. What can be happening?
You are processing way to many objects at the same time. You have two alternatives: split the processing (the dbedit commands file being processed) into smaller pieces, or open manually a dbedit session, and then copy-paste a reasonable amount of lines (200 or 250 are okay) at the same time. Wait until that is processed, and continue with the following ones…

Appendix B. Valid colors for objects in Object Filler

The following is the list of valid colors to be specified in the command line for Object Filler, as well as in the CSV file format.

aquamarine1	gold3
black	gray
blue	gray83
blue1	gray90
brown	green
burlywood4	lemonchiffon
coral	lightseagreen
cyan	lightskyblue4
darkorange3	magenta
darkseagreen3	medium
deepskyblue1	orange
dodgerblue3	pink
firebrick	red
Foreground	sienna
gold	yellow

When exporting objects with Object Dumper, due some limitations on the program, some times the color may be exported wrongly. On these cases, the object's color will be reset to black.

Appendix C. Default objects recognized by Object Dumper and Object Filler.

The following Object names are recognized as default (predefined) objects by Object Filler and Object Dumper. Names are NOT case-sensitive for the tools, which means that there's no difference between telnet, Telnet or TELNET.

In Object Filler, these objects are recognized as previously processed by –nopv switch while importing rules.

In Object Dumper, these objects will not be reported in a processing, unless the –d (default) switch is used.

IP Address Ranges

DAG_range

Dynamic Objects

LocalMachine
InternalNet
DMZNet
AuxiliaryNet

UDP Services

archie
biff
Blubster
bootp
Citrix_ICA_Browsing
CP_SecureAgent-udp
CU-SeeMe
daytime-udp
dhcp-rep-localmodule
dhcp-req-localmodule
Direct_Connect_UDP
discard-udp
domain-udp
E2ECP
echo-udp
eDonkey_4665
FreeTel-outgoing-server
FW1_load_agent
FW1_scv_keep_alive
FW1_snmp
GNUtella_rtr_UDP
GNUtella_UDP
GTPv0
GTPv1-C
GTPv1-U
H323_ras
H323_ras_only

TCP Services

AOL
AP-Defender
AT-Defender
Back_Door_Setup
Backage
BackDoor-G
Citrix_ICA
Connect-Back_Backdoor
ConnectedOnLine
CP_Exnet_PK
CP_Exnet_resolve
CP_redundant
CP_reporting
CP_rtm
CPD
CPD_amon
CPMI
CrackDown
CreativePartnerClnt
CreativePartnerSrvr
DaCryptic
DameWare
daytime-tcp
DerSphere
DerSphere_II
Direct_Connect_TCP
discard-tcp
domain-tcp
echo-tcp
eDonkey_4661
eDonkey_4662
Entrust-Admin
Entrust-KeyMgmt
exec
finger
Freak2k

HackaTack_31789
HackaTack_31791
Hotline_tracker
ICQ_locator
IKE
interphone
ISAKMP
Kerberos_v5_UDP
kerberos-udp
L2TP
MetaIP-UAT
microsoft-ds-udp
MSN_Messenger_1863_UDP
MSN_Messenger_5190
MSN_Messenger_Voice
MSSQL_resolver
MS-SQL-Monitor_UDP
MS-SQL-Server_UDP
name
nbdatagram
nbname
NEW-RADIUS
nfsd
NoBackO
ntp-udp
OnTime
pcANYWHERE-stat
RADIUS
RainWall_Daemon
RainWall_Status
RainWall_Stop
RDP
RexxRave
rip
RIPng
securid-udp
sip
sip_any
snmp
snmp-read
snmp-trap
Streamworks
SWTP_Gateway
SWTP_SMS
syslog
TACACS
tftp
time-udp
tunnel_test
udp-high-ports
vosaic-data

ftp
ftp-bidir
ftp-pasv
ftp-port
FW1
FW1_amon
FW1_clntauth_http
FW1_clntauth_telnet
FW1_CPRID
FW1_cvp
FW1_ela
FW1_ica_mgmt_tools
FW1_ica_pull
FW1_ica_push
FW1_ica_services
FW1_key
FW1_lea
FW1_log
FW1_mgmt
FW1_netso
FW1_omi
FW1_omi-sic
FW1_pslogon
FW1_pslogon_NG
FW1_sam
FW1_sds_logon
FW1_sds_logon_NG
FW1_snauth
FW1_topo
FW1_uaa
FW1_ufp
GateCrasher
GNUtella_rtr_TCP
GNUtella_TCP
gopher
GoToMyPC
H323
H323_any
HackaTack_31785
HackaTack_31787
HackaTack_31788
HackaTack_31790
HackaTack_31792
Hotline_client
http
https
ICKiller
ident
IKE_tcp
imap
iMesh

vosaic-ctrl
VPN1_IPSEC_encapsulation
WebTheater
who
WinMX
Yahoo_Messenger_Voice_Chat_UDP

Other Services
AH
backweb
egp
ESP
FreeTel-incoming
FreeTel-outgoing-client
ftp_mapped
FW1_Encapsulation
ggp
gre
gtp_path_mgmt
gtp_reverse
gtp_v0_path_mgmt
gtp_v1_path_mgmt
http_mapped
icmp-proto
igmp
igrp
ospf
rip-response
Sitara
SKIP
smtp_mapped
traceroute
tunnel_test_mapped
vrrp
X11-verify

ICMP Services
dest-unreach
echo-reply
echo-request
ICMP_frag_needed
info-reply
info-req
mask-reply
mask-request
param-prblm
redirect
source-quench
time-exceeded
timestamp
timestamp-reply

InCommand
IPSO_Clustering_Mgmt_Protocol
irc1
irc2
IS411-srvr
Jade
Kaos
KaZaA
Kazaa
Kerberos_v5_TCP
kerberos-tcp
Kuang2
ldap
ldap-ssl
login
lotus
lpdw0rm
Madster
microsoft-ds
Mneah
MSN_Messenger_File_Transfer
MSNP
MS-SQL-Monitor
MS-SQL-Server
Multidropper
Napster_Client_6600-6699
Napster_directory_4444
Napster_directory_5555
Napster_directory_6666
Napster_directory_7777
Napster_directory_8888_primary
Napster_redirector
nbsession
NCP
netshow
netstat
nfsd-tcp
nntp
ntp-tcp
OAS-NameServer
OAS-ORB
OpenWindows
Orbix-1570
Orbix-1571
pcANYWHERE-data
pcTELECOMMUTE-FileSync
pop-2
pop-3
Port_6667_trojans
pptp-tcp
RainWall_Command

RPC Services

cachefsd
cmsd
mountd
nfsprog
nisplus
nlockmgr
pcnfsd
rstat
rwall
sadmind
snmpXdmid
statd
ttdbserverd
ypbind
yppasswd
ypserv
ypupdated
ypxfrd

DCE-RPC Services

ALL_DCE_RPC
DCOM-RemoteActivation
HP-OpCctla
HP-OpCctla-bulk
HP-OpCctla-cfgpush
HP-OpCdistm
HP-OpCmsgrd-coa
HP-OpCmsgrd-m2m
HP-OpCmsgrd-std
MSExchangeADL
MSExchangeDirRef
MSExchangeDirRep
MSExchangeDSNSPI
MSExchangeDSRep
MSExchangeDSXDS
MSExchangeIS
MSExchangeMTA
MSExchangeStoreAdm
MSExchangeSysAtt
MSExchangeSysAttPriv
RAT
Real-Audio
RealSecure
Remote_Storm
rtsp
securidprop
Shadyshell
shell
smtp
SocketsdesTroie
sqlnet1
sqlnet2-1521
sqlnet2-1525
sqlnet2-1526
ssh
ssh_version_2
ssl_v3
StoneBeat-Control
StoneBeat-Daemon
SubSeven
T.120
TACACSplus
tcp-high-ports
telnet
Terrortrojan
TheFlu
time-tcp
TransScout
Trinoo
UltorsTrojan
uucp
wais
winframe
WinHole
X11
Xanadu
Yahoo_Messenger_messages
Yahoo_Messenger_Voice_Chat_TCP
Yahoo_Messenger_Webcams

Appendix D. Features Roadmap.

The following is a list of things that I think I'll include in the tools at some point of the time. They are not given in any particular order and there's no estimate on when they will be done. You may always send your suggestions and help prioritize this list by sending feedback.

Object Filler

- Support for security rules with User Groups as source, all the possible values in the track field and Resources within Services in CSV files
- Support for NAT rules in CSV Files and possibly Cisco PIX/Cisco Routers
- Support for NetScreen rules
- Support for SideWinder services
- Support for more file types from other firewall brands (?) – need sample files
- Support for a timestamp with created objects/rules
- Solaris native support via CLI
- Have a Windows and GNU/Linux native GUI
- Support different rulebases (names) from CSV files.
- Support colors with network object and service groups
- Support routes listing (netstat –nr) as a source file type for Object Filler
- Create a debug mode

Object Dumper

- Support for rules with User Groups as source, All the possible values in the track field and Resources within Services
- Support for NAT rules
- Support for a timestamp with dumper objects/rules
- Support for dumping rules only (without having to dump objects)
- Solaris native support via CLI
- Have a Windows and GNU/Linux native GUI
- Create a debug mode

Documentation

- Translate documentation to other Spanish and other languages. If there's people that volunteers for translating to other languages (or even Spanish ;-), such help would be always welcomed and acknowledged.
- Add images and screenshots, so the concepts may be clearer.

NAME

memtester − stress test to find memory subsystem faults.

SYNOPSIS

memtester [-p PHYSADDR [-d DEVICE]] <*MEMORY*> [*ITERATIONS*]

DESCRIPTION

memtester is an effective userspace tester for stress-testing the memory subsystem. It is very effective at finding intermittent and non-deterministic faults. Note that problems in other hardware areas (overheating CPU, out-of-specification power supply, etc.) can cause intermittent memory faults, so it is still up to you to determine where the fault lies through normal hardware diagnostic procedures; memtester just helps you determine whether a problem exists.

memtester will malloc(3) the amount of memory specified, if possible. If this fails, it will decrease the amount of memory requested until it succeeds. It will then attempt to mlock(3) this memory; if it cannot do so, testing will be slower and much less effective. Run memtester as root so that it can mlock the memory it tests.

Note that the maximum amount of memory that memtester can test will be less than the total amount of memory installed in the system; the operating system, libraries, and other system limits take some of the available memory. memtester is also limited to the amount of memory available to a single process; for example, on 32-bit machines with more than 4GB of memory, memtester is still limited to less than 4GB.

Note that it is up to you to know how much memory you can safely allocate for testing. If you attempt to allocate more memory than is available, memtester should figure that out, reduce the amount slightly, and try again. However, this can lead to memtester successfully allocating and mlocking essentially all free memory on the system -- if other programs are running, this can lead to excessive swapping and slowing the system down to the point that it is difficult to use. If the system allows allocation of more memory than is actually available (overcommit), it may lead to a deadlock, where the system halts. If the system has an out-of-memory process killer (like Linux), memtester or another process may be killed by the OOM killer.

So choose wisely.

OPTIONS

-p PHYSADDR

tells memtester to test a specific region of memory starting at physical address PHYSADDR (given in hex), by mmap(2)ing a device specified by the -d option (below, or /dev/mem by default). This is mostly of use to hardware developers, for testing memory-mapped I/O devices and similar. Note that the memory region will be overwritten during testing, so it is not safe to specify memory which is allocated for the system or for other applications; doing so will cause them to crash. If you absolutely must test a particular region of actual physical memory, arrange to have that memory allocated by your test software, and hold it in this allocated state, then run memtester on it with this option.

MEMORY

the amount of memory to allocate and test, in megabytes by default. You can include a suffix of B, K, M, or G to indicate bytes, kilobytes, megabytes, or gigabytes respectively.

ITERATIONS

(optional) number of loops to iterate through. Default is infinite.

ENVIRONMENT

If the environment variable MEMTESTER_TEST_MASK is set, memtester treats the value as a bitmask of which tests (other than the stuck address test) to run. The value can be specified in decimal, in octal (with a leading 0), or in hexadecimal (with a leading 0x). The specific bit values corresponding to particular tests may change from release to release; consult the list of tests in the source for the appropriate index values for the version of memtester you are running. Note that skipping some tests will reduce the time it takes for memtester to run, but also reduce memtester's effectiveness.

NOTE

memtester must be run with root privileges to mlock(3) its pages. Testing memory without locking the pages in place is mostly pointless and slow.

EXIT CODE

memtester's exit code is 0 when everything works properly. Otherwise, it is the logical OR of the following values:

x01 error allocating or locking memory, or invocation error

x02 error during stuck address test

x04 error during one of the other tests

AUTHOR

Written by Charles Cazabon.

REPORTING BUGS

Report bugs to <charlesc-memtester-bugs@pyropus.ca>.

COPYRIGHT

NAME

smtpclient -- simple SMTP client

SYNOPSIS

smtpclient [**−s** *STR*] [**−f** *ADDR*] [**−r** *ADDR*] [**−e** *ADDR*] [**−c** *ADDR*] [**−S** *HOST*] [**−P** *NUM*] [**−M**] [**−L**] [**−v**] *recipient ...*

smtpclient [**−V**] [**−h**]

VERSION

1.0.0 (21-10-1997)

DESCRIPTION

SMTPclient is a minimal SMTP client that takes an email message body and passes it on to a SMTP server (default is the MTA on the local host). Since it is completely self-supporting, it is especially suitable for use in restricted environments.

A typical usage is as a MTA emulating program on a dedicated webserver. There it is usually called from within a CGI program to receive a mail (constructed by the CGI program out of a HTML form) and to forward it to the actual mailserver.

OPTIONS

−s, **--subject=***STR*
: Specifies the `Subject:` header. This gives the message a title. Default is no subject.

−f, **--from=***ADDR*
: Specifies the `From:` address header. This is the logical senders address. Default is ‘‘`daemon`’’, which is probably wrong.

−r, **--reply-to=***ADDR*
: Specifies the `Reply-To:` address header. This is where replies should be send to. Default is no such header.

−e, **--errors-to=***ADDR*
: Specifies the `Errors-To:` address header. This is where delivery problems should be reported. Default is no such header.

−c, **--carbon-copy=***ADDR*
: Specifies the `Cc:` address header. This can contain one or more addresses (seperated by comma) to which one copy of the message is send to. Default is no such header.

−S, **--smtp-host=***HOST*
: Specifies the name or IP−address of the SMTP host to connect to. This is the host where the MTA is running to which the message is forwarded. By default, the mail is send to the SMTP daemon on `localhost` or to the host specified in the environment-variable `SMTPSERVER` if exists.

−P, **--smtp-port=***NUM*
: Specifies the port of the SMTP host to connect to. Default is port 25 (smtp).

−M, **--mime-encode**
: Use MIME−style translation to quoted-printable (base 16).

−L, **--use-syslog**
: Log errors to system’s *syslog* facility instead of *stderr*.

−v, **--verbose**
: Turn on verbose logging to *stdout*.

−V, **--version**
: Display the program version.

−h, **--help**
: Display the usage page.

RESULTS

The program terminates with a non-zero exit status in case of errors.

AUTHOR

Ralf S. Engelschall
rse@engelschall.com
www.engelschall.com

NAME

hping2 − send (almost) arbitrary TCP/IP packets to network hosts

SYNOPSIS

hping2 [**−hvnqVDzZ012WrfxykQbFSRPAUXYjJBuTG**] [**−c** *count*] [**−i** *wait*] [**−−fast**] [**−I** *interface*] [**−9** *signature*] [**−a** *host*] [**−t** *ttl*] [**−N** *ip id*] [**−H** *ip protocol*] [**−g** *fragoff*] [**−m** *mtu*] [**−o** *tos*] [**−C** *icmp type*] [**−K** *icmp code*] [**−s** *source port*] [**−p[+][+]** *dest port*] [**−w** *tcp window*] [**−O** *tcp offset*] [**−M** *tcp sequence number*] [**−L** *tcp ack*] [**−d** *data size*] [**−E** *filename*] [**−e** *signature*] [**−−icmp−ipver** *version*] [**−−icmp−iphlen** *length*] [**−−icmp−iplen** *length*] [**−−icmp−ipid** *id*] [**−−icmp−ipproto** *protocol*] [**−−icmp−cksum** *checksum*] [**−−icmp−ts**] [**−−icmp−addr**] [**−−tcpexitcode**] [**−−tcp-timestamp**] [**−−tr-stop**] [**−−tr-keep-ttl**] [**−−tr-no-rtt**] [**−−rand-dest**] [**−−rand-source**] hostname

DESCRIPTION

hping2 is a network tool able to send custom TCP/IP packets and to display target replies like ping program does with ICMP replies. hping2 handle fragmentation, arbitrary packets body and size and can be used in order to transfer files encapsulated under supported protocols. Using hping2 you are able to perform at least the following stuff:

- Test firewall rules
- Advanced port scanning
- Test net performance using different protocols, packet size, TOS (type of service) and fragmentation.
- Path MTU discovery
- Transferring files between even really fascist firewall rules.
- Traceroute-like under different protocols.
- Firewalk-like usage.
- Remote OS fingerprinting.
- TCP/IP stack auditing.
- A lot of others.

It's also a good didactic tool to learn TCP/IP. hping2 is developed and maintained by antirez@invece.org and is licensed under GPL version 2. Development is open so you can send me patches, suggestion and affronts without inhibitions.

HPING SITE

primary site at **http://www.hping.org**. You can found both the stable release and the instruction to download the latest source code at http://www.hping.org/download.html

BASE OPTIONS

-h --help

Show an help screen on standard output, so you can pipe to less.

-v --version

Show version information and API used to access to data link layer, *linux sock packet* or *libpcap.*

-c --count count

Stop after sending (and receiving) *count* response packets. After last packet was send hping2 wait COUNTREACHED_TIMEOUT seconds target host replies. You are able to tune COUNTREACHED_TIMEOUT editing hping2.h

-i --interval

Wait the specified number of seconds or micro seconds between sending each packet. --interval X set *wait* to X seconds, --interval uX set *wait* to X micro seconds. The default is to wait one second between each packet. Using hping2 to transfer files tune this option is really important in order to increase transfer rate. Even using hping2 to perform idle/spoofing scanning you should tune this option, see **HPING2-HOWTO** for more information.

--fast Alias for -i u10000. Hping will send 10 packets for second.

--faster Alias for -i u1. Faster then --fast ;) (but not as fast as your computer can send packets due to the signal-driven design).

-n --numeric
Numeric output only, No attempt will be made to lookup symbolic names for host addresses.

-q --quiet
Quiet output. Nothing is displayed except the summary lines at startup time and when finished.

-I --interface interface name
By default on linux and BSD systems hping2 uses default routing interface. In other systems or when there is no default route hping2 uses the first non-loopback interface. However you are able to force hping2 to use the interface you need using this option. Note: you don't need to specify the whole name, for example -I et will match eth0 ethernet0 myet1 et cetera. If no interfaces match hping2 will try to use lo.

-V --verbose
Enable verbose output. TCP replies will be shown as follows:

len=46 ip=192.168.1.1 flags=RA DF seq=0 ttl=255 id=0 win=0 rtt=0.4 ms tos=0 iplen=40 seq=0 ack=1380893504 sum=2010 urp=0

-D --debug
Enable debug mode, it's useful when you experience some problem with hping2. When debug mode is enabled you will get more information about **interface detection, data link layer access, interface settings, options parsing, fragmentation, HCMP protocol** and other stuff.

-z --bind
Bind CTRL+Z to **time to live (TTL)** so you will able to increment/decrement ttl of outgoing packets pressing CTRL+Z once or twice.

-Z --unbind
Unbind CTRL+Z so you will able to stop hping2.

PROTOCOL SELECTION

Default protocol is TCP, by default hping2 will send tcp headers to target host's port 0 with a winsize of 64 without any tcp flag on. Often this is the best way to do an 'hide ping', useful when target is behind a firewall that drop ICMP. Moreover a tcp null-flag to port 0 has a good probability of not being logged.

-0 --rawip
RAW IP mode, in this mode hping2 will send IP header with data appended with --signature and/or --file, see also --ipproto that allows you to set the ip protocol field.

-1 --icmp
ICMP mode, by default hping2 will send ICMP echo-request, you can set other ICMP type/code using **--icmptype --icmpcode** options.

-2 --udp
UDP mode, by default hping2 will send udp to target host's port 0. UDP header tunable options are the following: **--baseport, --destport, --keep.**

-8 --scan
Scan mode, the option expects an argument that describes groups of ports to scan. port groups are comma separated: a number describes just a single port, so 1,2,3 means port 1, 2 and 3. ranges are specified using a start-end notation, like 1-1000, that tell hping to scan ports between 1 and 1000 (included). the special word **all** is an alias for 0-65535, while the special word **known** includes all the ports listed in /etc/services.
Groups can be combined, so the following command line will scan ports between 1 and 1000 AND port 8888 AND ports listed in /etc/services: **hping --scan 1-1000,8888,known -S target.host.com**

Groups can be negated (subtracted) using a ! character as prefix, so the following command line will scan all the ports NOT listed in /etc/services in the range 1-1024: **hping --scan '1-1024,!known' -S target.host.com**

Keep in mind that while hping seems much more like a port scanner in this mode, most of the hping switches are still honored, so for example to perform a SYN scan you need to specify the **-S** option, you can change the TCP windows size, TTL, control the IP fragmentation as usually, and so on. The only real difference is that the standard hping behaviors are encapsulated into a scanning algorithm.

Tech note: The scan mode uses a two-processes design, with shared memory for synchronization. The scanning algorithm is still not optimal, but already quite fast.

Hint: unlike most scanners, hping shows some interesting info about received packets, the IP ID, TCP win, TTL, and so on, don't forget to look at this additional information when you perform a scan! Sometimes they shows interesting details.

-9 --listen signature

HPING2 listen mode, using this option hping2 waits for packet that contain *signature* and dump from *signature* end to packet's end. For example if hping2 --listen TEST reads a packet that contain **234-09sdflkjs45-TESThello_world** it will display **hello_world**.

IP RELATED OPTIONS

-a --spoof hostname

Use this option in order to set a fake IP source address, this option ensures that target will not gain your real address. However replies will be sent to spoofed address, so you will can't see them. In order to see how it's possible to perform spoofed/idle scanning see the **HPING2-HOWTO**.

--rand-source

This option enables the **random source mode**. hping will send packets with random source address. It is interesting to use this option to stress firewall state tables, and other per-ip basis dynamic tables inside the TCP/IP stacks and firewall software.

--rand-dest

This option enables the **random destination mode**. hping will send the packets to random addresses obtained following the rule you specify as the target host. You need to specify a numerical IP address as target host like **10.0.0.x**. All the occurrences of **x** will be replaced with a random number in the range 0-255. So to obtain Internet IP addresses in the whole IPv4 space use something like **hping x.x.x.x --rand-dest**. If you are not sure about what kind of addresses your rule is generating try to use the **--debug** switch to display every new destination address generated. When this option is turned on, matching packets will be accept from all the destinations.

Warning: when this option is enabled hping can't detect the right outgoing interface for the packets, so you should use the **--interface** option to select the desired outgoing interface.

-t --ttl time to live

Using this option you can set **TTL (time to live)** of outgoing packets, it's likely that you will use this with **--traceroute** or **--bind** options. If in doubt try '**hping2 some.host.com -t 1 --traceroute**'.

-N --id Set ip->id field. Default id is random but if fragmentation is turned on and id isn't specified it will be **getpid() & 0xFF**, to implement a better solution is in TODO list.

-H --ipproto

Set the ip protocol in RAW IP mode.

-W --winid

id from Windows* systems before Win2k has different byte ordering, if this option is enable hping2 will properly display id replies from those Windows.

-r --rel Display id increments instead of id. See the **HPING2-HOWTO** for more information. Increments aren't computed as id[N]-id[N-1] but using packet loss compensation. See relid.c for more information.

-f --frag

Split packets in more fragments, this may be useful in order to test IP stacks fragmentation performance and to test if some packet filter is so weak that can be passed using tiny fragments (anachronistic). Default 'virtual mtu' is 16 bytes. see also *--mtu* option.

-x --morefrag

Set more fragments IP flag, use this option if you want that target host send an **ICMP time-exceeded during reassembly**.

-y --dontfrag

Set don't fragment IP flag, this can be used to perform **MTU path discovery**.

-g --fragoff fragment offset value

Set the fragment offset.

-m --mtu mtu value

Set different 'virtual mtu' than 16 when fragmentation is enabled. If packets size is greater that 'virtual mtu' fragmentation is automatically turned on.

-o --tos hex_tos

Set **Type Of Service (TOS)**, for more information try **--tos help**.

-G --rroute

Record route. Includes the RECORD_ROUTE option in each packet sent and displays the route buffer of returned packets. Note that the IP header is only large enough for nine such routes. Many hosts ignore or discard this option. Also note that using hping you are able to use record route even if target host filter ICMP. Record route is an IP option, not an ICMP option, so you can use record route option even in TCP and UDP mode.

ICMP RELATED OPTIONS

-C --icmptype type

Set icmp type, default is **ICMP echo request** (implies --icmp).

-K --icmpcode code

Set icmp code, default is 0 (implies --icmp).

--icmp-ipver

Set IP version of IP header contained into ICMP data, default is 4.

--icmp-iphlen

Set IP header length of IP header contained into ICMP data, default is 5 (5 words of 32 bits).

--icmp-iplen

Set IP packet length of IP header contained into ICMP data, default is the real length.

--icmp-ipid

Set IP id of IP header contained into ICMP data, default is random.

--icmp-ipproto

Set IP protocol of IP header contained into ICMP data, default is TCP.

--icmp-cksum

Set ICMP checksum, for default is the valid checksum.

--icmp-ts

Alias for --icmptype 13 (to send ICMP timestamp requests).

--icmp-addr

Alias for --icmptype 17 (to send ICMP address mask requests).

TCP/UDP RELATED OPTIONS

-s --baseport source port

hping2 uses source port in order to guess replies sequence number. It starts with a base source port number, and increase this number for each packet sent. When packet is received sequence number can be computed as *replies.dest.port - base.source.port*. Default base source port is random, using

this option you are able to set different number. If you need that source port not be increased for each sent packet use the *-k --keep* option.

-p --destport [+][+]dest port
Set destination port, default is 0. If '+' character precedes dest port number (i.e. +1024) destination port will be increased for each reply received. If double '+' precedes dest port number (i.e. ++1024), destination port will be increased for each packet sent. By default destination port can be modified interactively using **CTRL+z**.

--keep keep still source port, see *--baseport* for more information.

-w --win
Set TCP window size. Default is 64.

-O --tcpoff
Set fake tcp data offset. Normal data offset is tcphdrlen / 4.

-M --tcpseq
Set the TCP sequence number.

-L --tcpack
Set the TCP ack.

-Q --seqnum
This option can be used in order to collect sequence numbers generated by target host. This can be useful when you need to analyze whether TCP sequence number is predictable. Output example:

#hping2 win98 --seqnum -p 139 -S -i u1 -I eth0
HPING uaz (eth0 192.168.4.41): S set, 40 headers + 0 data bytes
2361294848 +2361294848
2411626496 +50331648
2545844224 +134217728
2713616384 +167772160
2881388544 +167772160
3049160704 +167772160
3216932864 +167772160
3384705024 +167772160
3552477184 +167772160
3720249344 +167772160
3888021504 +167772160
4055793664 +167772160
4223565824 +167772160

The first column reports the sequence number, the second difference between current and last sequence number. As you can see target host's sequence numbers are predictable.

-b --badcksum
Send packets with a bad UDP/TCP checksum.

--tcp-timestamp
Enable the TCP timestamp option, and try to guess the timestamp update frequency and the remote system uptime.

-F --fin Set FIN tcp flag.

-S --syn
Set SYN tcp flag.

-R --rst Set RST tcp flag.

-P --push
Set PUSH tcp flag.

-A --ack
Set ACK tcp flag.

-U --urg
Set URG tcp flag.

-X --xmas
Set Xmas tcp flag.

-Y --ymas
Set Ymas tcp flag.

COMMON OPTIONS

-d --data data size
Set packet body size. Warning, using --data 40 hping2 will not generate 0 byte packets but protocol_header+40 bytes. hping2 will display packet size information as first line output, like this: **HPING www.yahoo.com (ppp0 204.71.200.67): NO FLAGS are set, 40 headers + 40 data bytes**

-E --file filename
Use **filename** contents to fill packet's data.

-e --sign signature
Fill first *signature length* bytes of data with *signature*. If the *signature length* is bigger than data size an error message will be displayed. If you don't specify the data size hping will use the signature size as data size. This option can be used safely with *--file filename* option, remainder data space will be filled using *filename*.

-j --dump
Dump received packets in hex.

-J --print
Dump received packets' printable characters.

-B --safe
Enable safe protocol, using this option lost packets in file transfers will be resent. For example in order to send file /etc/passwd from host A to host B you may use the following:
[host_a]
 # hping2 host_b --udp -p 53 -d 100 --sign signature --safe --file /etc/passwd
[host_b]
 # hping2 host_a --listen signature --safe --icmp

-u --end
If you are using *--file filename* option, tell you when EOF has been reached. Moreover prevent that other end accept more packets. Please, for more information see the **HPING2-HOWTO**.

-T --traceroute
Traceroute mode. Using this option hping2 will increase ttl for each **ICMP time to live 0 during transit** received. Try **hping2 host --traceroute**. This option implies --bind and --ttl 1. You can override the ttl of 1 using the --ttl option. Since 2.0.0 stable it prints RTT information.

--tr-keep-ttl
Keep the TTL fixed in traceroute mode, so you can monitor just one hop in the route. For example, to monitor how the 5th hop changes or how its RTT changes you can try **hping2 host --traceroute --ttl 5 --tr-keep-ttl**.

--tr-stop
If this option is specified hping will exit once the first packet that isn't an ICMP time exceeded is received. This better emulates the traceroute behavior.

--tr-no-rtt

Don't show RTT information in traceroute mode. The ICMP time exceeded RTT information aren't even calculated if this option is set.

--tcpexitcode

Exit with last received packet tcp->th_flag as exit code. Useful for scripts that need, for example, to known if the port 999 of some host reply with SYN/ACK or with RST in response to SYN, i.e. the service is up or down.

TCP OUTPUT FORMAT

The standard TCP output format is the following:

len=46 ip=192.168.1.1 flags=RA DF seq=0 ttl=255 id=0 win=0 rtt=0.4 ms

len is the size, in bytes, of the data captured from the data link layer excluding the data link header size. This may not match the IP datagram size due to low level transport layer padding.

ip is the source ip address.

flags are the TCP flags, R for RESET, S for SYN, A for ACK, F for FIN, P for PUSH, U for URGENT, X for not standard 0x40, Y for not standard 0x80.

If the reply contains **DF** the IP header has the don't fragment bit set.

seq is the sequence number of the packet, obtained using the source port for TCP/UDP packets, the sequence field for ICMP packets.

id is the IP ID field.

win is the TCP window size.

rtt is the round trip time in milliseconds.

If you run hping using the **-V** command line switch it will display additional information about the packet, example:

len=46 ip=192.168.1.1 flags=RA DF seq=0 ttl=255 id=0 win=0 rtt=0.4 ms tos=0 iplen=40 seq=0 ack=1223672061 sum=e61d urp=0

tos is the type of service field of the IP header.

iplen is the IP total len field.

seq and ack are the sequence and acknowledge 32bit numbers in the TCP header.

sum is the TCP header checksum value.

urp is the TCP urgent pointer value.

UDP OUTPUT FORMAT

The standard output format is:

len=46 ip=192.168.1.1 seq=0 ttl=64 id=0 rtt=6.0 ms

The field meaning is just the same as the TCP output meaning of the same fields.

ICMP OUTPUT FORMAT

An example of ICMP output is:

ICMP Port Unreachable from ip=192.168.1.1 name=nano.marmoc.net

It is very simple to understand. It starts with the string "ICMP" followed by the description of the ICMP error, Port Unreachable in the example. The ip field is the IP source address of the IP datagram containing the ICMP error, the name field is just the numerical address resolved to a name (a dns PTR request) or UNKNOWN if the resolution failed.

The ICMP Time exceeded during transit or reassembly format is a bit different:

TTL 0 during transit from ip=192.168.1.1 name=nano.marmoc.net

TTL 0 during reassembly from ip=192.70.106.25 name=UNKNOWN

The only difference is the description of the error, it starts with TTL 0.

AUTHOR

Salvatore Sanfilippo <antirez@invece.org>, with the help of the people mentioned in AUTHORS file and at http://www.hping.org/authors.html

BUGS

Even using the --end and --safe options to transfer files the final packet will be padded with 0x00 bytes.

Data is read without care about alignment, but alignment is enforced in the data structures. This will not be a problem under i386 but, while usually the TCP/IP headers are naturally aligned, may create problems with different processors and bogus packets if there is some unaligned access around the code (hopefully none).

On solaris hping does not work on the loopback interface. This seems a solaris problem, as stated in the tcpdump-workers mailing list, so the libpcap can’t do nothing to handle it properly.

SEE ALSO

ping(8), traceroute(8), ifconfig(8), nmap(1)

NAME

links − lynx-like alternative character mode WWW browser

SYNOPSIS

links *[options] URL*

DESCRIPTION

links is a text mode WWW browser, supporting colors, correct table rendering, background downloading, menu driven configuration interface and slim code.

Frames are supported. You can have different file formats associated with external viewers. mailto: and telnet: are supported via external clients.

links can handle local (file://) or remote (http://, ftp:// or https:// if there's compiled-in SSL support) URLs.

OPTIONS

Most options can be set in the user interface or config file, so usually you do not need to care about them.

-anonymous
Restrict links so that it can run on an anonymous account. No local file browsing. No downloads. Executing of viewers is allowed, but user can't add or modify entries in association table.

-assume-codepage *<codepage>*
Use the given codepage when the webpage did not specify its codepage. (default: ISO 8859-1)

-async-dns *<0>/<1>*
Asynchronous DNS resolver on(1)/off(0). Default is on.

-base-session *<number>*
Linkses are connecting together, forming sessions by default. Normally, there's only one session with number 0. In this way, you can make him to form more sessions at once. See also **-no-connect** option.

-download-dir *<path>*
Default download directory. (default: actual dir)

-download-utime *<0>/<1>*
Set time of download files on(1)/off(0). Default is off.

-dump Write a plain-text version of the given HTML document to stdout.

-format-cache-size *<num>*
Number of formatted document pages cached. (default: 5)

-ftp.anonymous-password *<password>*
Use this as the password for anonymous ftp sites.

-ftp-proxy <*host:port*>
Host and port number of the FTP proxy, or blank. (default: blank)

-help Print a help screen.

-http-bugs.http10 <*0*>/<*1*>
Turn use of HTTP/1.0 on(1)/off(0). Default is off.

-http-bugs.allow-blacklist <*0*>/<*1*>
Turn blacklist of buggy servers on(1)/off(0). Default is on.

-http-bugs.bug-302-redirect <*0*>/<*1*>
Turn broken redirect 302 (violates RFC, but compatible with Netscape) on(1)/off(0). Default is off.

-http-bugs.bug_post-no-keepalive <*0*>/<*1*>
Turn no keepalive connection after POST request (for some buggy PHP databases) on(1)/off(0). Default is off.

-http-proxy <*host:port*>
Host and port number of the HTTP proxy, or blank. (default: blank)

-language <*langname*>
Language to use in user interface.

-max-connections <*max*>
Maximum number of concurrent connections. (default: 10)

-max-connections-to-host <*max*>
Maximum number of concurrent connection to a given host. (default: 2)

-memory-cache-size <*Kbytes*>
Cache memory in Kilobytes. (default: 1024)

-width <*size*>
Size of screen in characters, used in combination with -dump. (default: 80)

-no-connect
Runs links as a separate instance - instead of connecting to existing instance. Prevents him to connecting to an existing session. See also **-base-session** option.

-receive-timeout <*sec*>
Timeout on receive. (default: 120)

-retries <*retry*>
Number of retries. (default: 3)

-source Write the given HTML document in source form to stdout.

-unrestartable-receive-timeout <*sec*>
Timeout on non restartable connections. (default: 600)

-version
Print the *links* version number and exit.

NAVIGATION KEYS

You may optionally prefix each of this keys with a number, telling its repeat count (how much times to do it). You can also re-bind those keys, altough this feature is still in the state of testing, so undocumented. The keys you may use for navigation are:

PGDN page down

Space page down

PGUP page up

b page up

B page up

CursorDOWN
next link/down

CursorUP
prev link/up

^INS copy to clipboard

^C copy to clipboard

INS scroll up

^P scroll up

DEL scroll down

^N scroll down

[scroll left

] scroll right

HOME
home

END end of page

CursorRIGHT
enter link/press button

ENTER
enter link/press button

CursorLEFT
go back

d download link

D download link

/ search in the page

? search back in the page

n find next match

N find next match backwards

f zoom actual frame

F zoom actual frame

^R reload page

g go to URL

G go to the current URL

a add a new bookmark

A add a new bookmark

s bookmark manager

S bookmark manager

q quit

Q quit

= document information

| header information

**** toggle HTML source/rendered view

***** toggle display of images

TAB next frame

ESC menu/escape

F9 menu

F10 file menu

EDITING KEYS

The following keys can be used while editing a line/jumping to a URL:

CursorRIGHT
move right

CursorLEFT
move left

HOME
jump to the beginning

^A jump to the beginning

END jump to the end

^E jump to the end

^INS copy to clipboard

^B copy to clipboard

^X cut to clipboard

^V paste from clipboard

ENTER
enter line

BACKSPACE
delete back character

^H delete back character

DEL delete character

^D delete character

^U delete from beginning of the line

^K delete to the end of the line

^W auto complete line

ENVIRONMENT

CONFIG_DIR
: The location of ".links/"

WWW_HOME
: Homepage location (as in **lynx**)

LINKS_XTERM
: The command to run when selecting "File/New window" and if **DISPLAY** is defined (default "xterm -e")

LINKS_TWTERM
: The command to run when selecting "File/New window" and if **TWDISPLAY** is defined (default "twterm -e")

SHELL
: Used for "File/OS Shell" menu

COMSPEC
: Used for "File/OS Shell" menu in DOS/Windows

FILES

~/.links/links.cfg
: Per-user config file, automatically created by **links.** Its format is same as the one of *user.cfg.*

~/.links/html.cfg
: Per-user config file, automatically created by **links.** It contains HTML rendering options. Its format is same as the one of *user.cfg.*

~/.links/user.cfg
: Per-user config file, not overwritten while saving **links** configuration. You can add your own options there (e.g. keybindings). Each line consists from option name (same as the command-line one, but without leading dash and with dashes converted to underscores), space, and its value, in quotes if it’s a string. Lines which start with # are considered as comments and skipped.

~/.links/bookmarks
Bookmarks file

~/.links/links.his
Sites history file

~/.links/socket
Internal links socket for communication between its instances.

PLATFORMS

links is known to work on Linux, FreeBSD, OpenBSD, Solaris, IRIX, HPUX, Digital Unix, AIX, OS/2 and BeOS. Port for Win32 is in state of beta testing.

BUGS

Can't connect to some FTP servers (Novell, NT). Connection stays in "Request sent" state.

Frames don't work if there're more frames with same name. Turn them off in such case.

You can't upload large files; it takes _lots_ of memory.

You shouldn't press '^Z' when you are in a viewer

Please report any other bugs you find to **Mikulas Patocka <mikulas@artax.karlin.mff.cuni.cz>** or to **links mailing list <links-list@linuxfromscratch.org>.**

LICENSE

links is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

AUTHOR

links was written by **Mikulas Patocka <mikulas@artax.karlin.mff.cuni.cz>.** See file *AUTHORS* in the source tree for a list of people contributing to this project.

The homepage of links can be found at **http://links.browser.org/**

This manual page was written by Peter Gervai <grin@tolna.net>, using excerpts from a (yet?) unknown *links* fan for the Debian GNU/Linux system (but may be used by others). Contributions from Francis A. Holop. Extended, clarified and made more up-to-date by Petr Baudis <pasky@ji.cz>.

SEE ALSO

lynx(1), **w3m**(1), **wget**(1)

NAME

edge_backup.sh − create RCS based archives for edge configurations

PACKAGE

UNItools

SYNOPSIS

edge_backup.sh [-v]

DESCRIPTION

`edge_backup.sh(1)` is a poor mans backup program for SofaWare edge Appliances managed through a service centre.

`edge_backup.sh(1)` is intented to run periodically from `cron(1)` on the *service centre* which also might host the *managemet center*. The *internal IP address* of the edge is reachable through the VPN tunnel.

The script is intented to run from `cron(1)` as the user `admin` on the *service center*.

The recommended `crontab(5)` is shown below:

```
minutes  hours  day-of-month  month  day-of-week command
00 7-23 * * * /var/opt/UNItools/bin/edge_backup.sh >/dev/null 2>&
```

OPTIONS

`-v` Print diagnostic output to STDOUT.

FILES

The config file should be located in `$HOME/etc/edge_backup.cfg`.
The config file is a shell script containing one variable:

```
LIST_OF_INSIDE_IPADDRS="
#
# carefull here be dragons
#
# ipv4addr     login_user      login_passwd    edge_admin_port
#
172.28.11.1    admin           1qazxsw2        981
"
```

The format is shown above. Blank lines and linies beginning with # are ignored.

COMMANDS

`bash(1)`, `sed(1)`, `wget(1)`, etc.

SEE ALSO

The Wiki at http://info.ssi.uni-c.dk.

DIAGNOSTICS

Diagnostic putput is written to syslog. No further information is provided in case of error(s).

KNOWN BUGS AND LIMITATIONS

IP addresses for edge appliances must be maintained both in the
object database and the configuration file.

VERSION

$Date: 2011/09/07 11:27:54 $
$Revision: 1.2 $
$Source: /lan/ssi/projects/UNItools/src/SPLAT/NGX/pkgs/edge_backup/edge_backup-1.0/edge_backup.sh,v $
$State: Exp $

HISTORY

See rlog $Source: /lan/ssi/projects/UNItools/src/SPLAT/NGX/pkgs/edge_backup/edge_backup-1.0/edge_backup.sh,v $

AUTHOR(S)

Niels Thomas Haugård
E-mail: niels.thomas.haugaard@uni-c.dk
UNI•C
Professional Security and Service
Danish Technical University, bldg 304
DK-2800 Lyngby, Denmark

Name

di − disk information

Synopsis

di [**−AaghHklLmnPtZ**] [**−b** *block−size*] [**−d** *display−size*] [**−f** *format*] [**−I** *include−fstyp−list*] [**−s** *sort−type*] [**−w** *block−width*] [**−W** *inode−width*] [**−x** *exclude−fstyp−list*] [**−X** *debug-level*] [**−z** *zone−name*] [*file* [...]]

mi

Description

di Displays usage information on mounted filesystems. Block values are reported in megabytes (1024k bytes) by default, but may be changed. If *file* is specified, the usage information for the partition on which *file* is located is printed.

Unless the −a flag is specified, the following mounted filesystems will not normally be displayed: filesystems with total blocks <= 0; filesystems marked by the operating system as "ignore"; automounted filesystems that are duplicates of other normally mounted filesystems; loopback filesystems that are part of a zone (Solaris).

mi Displays the mounted filesystem information.

Several switches may be specified to control the output of *di* and *mi*:

−A Print all fields (used for debugging). Mount points and special device names are printed at full width. The −a flag is set.

−a Prints all mounted devices (normally, those with 0 total blocks are not printed − e.g. **/dev/proc**, **/dev/fd**).

−b *block−size* (compatibility: −B)
Change the base block size from 1024 (default) to the size specified. *block−size* may be one of: k − 1024 bytes, d − 1000 bytes, or a specific size.

−d *display−size* Display the blocks in units specified by *display−size*. *display−size* may be one of: 512 − POSIX, k − kilobytes, m − megabytes, g − gigabytes, t − terabytes, P − petabytes, E − exabytes, Z − zettabytes, Y − yottabytes, h − Scaled alternative 1, H − Scaled alternative 2, or a specific value to use as the block size.

Block display sizes greater than 1024 bytes are displayed with a precision of one decimal place after the radix.

The Scaled alternatives scale the sizes displayed and appends a suffix (e.g. 48.0k, 3.4M).

With scaled alternative 1, sizes within a line may scale to different units.

Scaled alternative 2 scales all the sizes in each individual line to the same unit size (the largest needed).

−f *format* Use the specified format string *format*. Format strings are described below.

−g (alias for: −dg)
Display sizes in gigabytes.

−h (alias for: −dh)
Display partition sizes in scaled alternative 1 format.

−H (alias for: −dH)
Display partition sizes in scaled alternative 2 format.

−I *include−fstype−list* (compatibility: −F)
Include *only* the file system types listed in *include−fstyp−list*. The list is a comma separated list of file system types. Multiple −I options may be specified. e.g. −I nfs,rfs or −I nfs −I rfs.

−k (alias for: −dk)
Display sizes in Kbytes.

−l Display only local file systems.

−L Don't do the check for duplicate filesystems.

−m (alias for: −dm)
Display sizes in megabytes.

−n Do not print a header line above the list of file systems. Useful when parsing the output of *di*.

−P Output format is POSIX standard. 512 byte block size is the default. The −k option may be used.

−s *sort−type*
Use *sort−type* to sort the output. The output of *di* is normally sorted by name. The output may be left unsorted (**-s n** command line switch) i.e. as it appears in the mount table or may be sorted by the special device name (**-s s** command line switch). If the **-s r** command line switch is used, the sort order will be reversed. If *file* is specified on the command line, no sorting is done.

−t Print a totals line below the list of file systems. Memory filesystems (tmpfs, mfs) and read-only filesystems are not added to the totals.

−w *block−width*
Set the print width for block values. Default is eight.

−W *inode−width*
Set the print width for inode values. Default is seven.

−x *exclude−fstype−list* (old: −i)
Exclude the file system types listed in *exclude−fstyp−list*. The list is a comma separated list of file system types. Multiple −x options may be specified. e.g. −x nfs,rfs or −x nfs −x rfs.

−X *level*
Set the program's debugging level to *debug-level*.

−z *zone-name*
Display the filesystems for the specified zone. The zone must be visible to the user.

−Z (alias for: −z all)
Display the filesystems for all visible zones.

Format strings

The output of *di* may be specified via a format string. This string may be given either via the **-f** command line switch or by the **DIFMT** environment variable. The format string may specify the following columns:

m Print the name of the mount point.

M Print the name of the mount point, at full length. The mount point is formatted to the maximum width necessary for the longest mount point name.

b Print the total number of megabytes on the file system. See also the −d option.

B Print the total number of megabytes on the file system available for use by normal users. See also the −d option.

u Print the number of megabytes in use on the file system (actual number of megabytes used = total − free). See also the −d option.

c Print the number of megabytes not available for use by normal users (total − available). See also the −d option.

f Print the number of free (unused) megabytes on the file system. See also the −d option.

v Print the number of megabytes available for use by normal users. See also the −d option.

p Print the percentage of megabytes not available for use by normal users (number of megabytes not available for use / total disk space). See also the −d option.

1 Print the percentage of total megabytes in use (actual number of megabytes used / total disk space). See also the −d option.

2 Print the percentage of megabytes in use, BSD-style. Represents the percentage of user-available space in use. Note that values over 100% are possible (actual number of megabytes used / disk space available to non-root users). See also the −d option.

a Print the percentage of megabytes available for use by normal users (number of megabytes available for use / total disk space). See also the −d option.

3 Print the percentage of total megabytes free (actual number of megabytes free / total disk space). See also the −d option.

i Print the total number of file slots (inodes) that can be created on the file system.

U Print the number of file slots in use.

F Print the number of file slots available.

P Print the percentage of file slots in use.

s Print the file system name (special device or remote mount point).

S Print the file system name (special device or remote mount point), at full length. The file system name is formatted to the maximum width necessary for the longest file system name.

t Print the file system type.

T Print the file system type at full length. The file system type is formatted to the maximum width necessary for the longest file system type.

I Print the time the filesystem was mounted. This column is not supported on all systems.

O Print the filesystem mount options.

The default format string for *di* is **smbuvpT**.

The default format string for *mi* is **MSTIO**.

The format string may also contain any other character not listed above. The character will be printed as is. e.g. di −f 'mbuvp|iUFP' will print the character '|' between the disk usage and the file slot usage. The command sequence (Bourne Shell):

```
di −f 'mbuvp
miUFP'
```

will print two lines of data for each filesystem.

Examples

Various *df* equivalent format strings for System V release 4 are:

/usr/bin/df −v	di −P −f msbuf1
/usr/bin/df −k	di −dk −f sbcvpm
/usr/ucb/df	di −dk −f sbuv2m

If you like your numbers to add up/calculate the percentage correctly, try one of the following format strings:

```
di −f SMbuf1T
di −f SMbcvpT
di −f SMBuv2T
```

Environment Variables

The DIFMT environment variable may be used to specify the default display format string.

The DI_ARGS environment variable may be used to specify command line arguments. e.g. If you always want gigabytes displayed, set DI_ARGS equal to "-dg". Any command line arguments specified will override the DI_ARGS environment variable.

The GNU df POSIXLY_CORRECT, and DF_BLOCK_SIZE and the BSD BLOCKSIZE environment variables are honored.

Note

For filesystems that do not report available blocks (e.g. System V release 3), the number of available blocks is considered to be the number of free blocks.

WARNING

Do not replace your system's *df* command with this program. You will in all likelihood break your installation procedures.

See Also

df(1), fstab(5), getmnt(2), getmntinfo(2), mnttab(4), mount(1M) statfs(2), statvfs(2)

Bugs

Send bug reports to: di-bugs@gentoo.com

Website

http://www.gentoo.com/di/

Author

Brad Lanam, Walnut Creek, CA (bll@gentoo.com)