FORSCHUNGSZENTRUM JÜLICH GmbH KFA

Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61–6402

Informationszentrum, Tel. (02461) 61–6658

Referenzkarte

KFA-ZAM-RFK-0009

M. Sczimarowsky

13.2.95

UNIX

UNIX Commands: Online Help (man pages)

The execution of almost every UNIX command can be controlled or modified by the use of options. For shortening reasons options are not generally mentioned on this card. Online information can be required by the following commands:

man *command*	information about the command *command*
man -k *searchstring*	display one-line synopsis of **man page(s)** that contain keyword *searchstring*

Command Line Editing (Korn Shell)

<ESC> k [k] [j]	Scroll shell history to retrieve commands that were executed in the past. **k**: last recent command, **j**: next recent command. If problems occur within a remote terminal session, enter **set -o vi**

Cursor Positioning (activate Command Mode by pressing <ESC>)

<h> (**<l>**)	cursor left (right)
<w> (**<b>**)	move cursor to beginning of next (previous) word
<x>	delete character (at cursor position)
<D>	erase end of line

Input Mode (execute the following commands in command mode)

<i> | **<a>**	insert text before | after cursorposition
<r>	replace character
<R>	replace text starting at cursor position
<A>	append text at end of line

For more information: see vi reference card (KFA-ZAM-RFK-0010)

Working environment

passwd	change password (eventually following system dependent rules)
logname	login name
id	login name and active group id
groups	display group membership
env	display values of environment variables

File Operations

The position of a file or directory within the file system hierarchy may be specified in an absolute or relative manner (starting point root directory **/** or actual directory **.**)

cat *file1* **[***file2* **...]**	(concatenate) write *file1[, file2 ,...]* to stdout
more *file1* **[***file2* **...]**	display *file1[, file2 ,...]* one screenful at a time
pg *file1* **[***file2* **...]**	see **more**
touch *file1* **[***file2* **...]**	Update file access and modification times. If not existent, an empty *file1* is created
rm *file1* **[***file2* **...]**	remove *file1*[, *file2*, ...]
cp *file1 file2*	Copy files (source: *file1*, destination: *file2*). If *file2* exists, it will be overwritten (if permitted)
cp *file1* **[***file2*..**] dir**	Copy files to directory **dir**
mv *file1 file2*	Rename *file1*, new name: *file2*. If *file2* exists, it will be overwritten (if permitted)
mv *file1* **[***file2*..**] dir**	Move *file1* [*file2*] to directory *dir*
diff *file1 file2*	Textual comparison of *file1* and *file2*. Display differing lines
cmp *file1 file2*	Byte-by-byte comparison of *file1* and *file2*. Display differing characters
compress *file*	Compress data of *file*, the result is written to *file*.Z. To expand such data, use **uncompress** *file*.Z.
find *path crit action*	Search for files recursively starting at *path*. Find files matching *crit* and pass their names to *action*

Directory Operations

pwd	print working directory (current position in directory tree)
ls *[dir]*	list contents of directory *dir*. If *dir* is missing, list working directory (example: **ls -lisa**)
du *[-k] [dir]*	display disk space in use for *dir*. Option *-k*: list in 1 KB units.

cd *[dir]*	change current directory to *dir*. If *dir* is missing, change to homedirectory. **cd ..** changes to parent directory
mkdir *dir1 [dir2 ...]*	create new directory *dir1[, dir2 ...]*.
rmdir *dir1 [dir2 ...]*	remove directory *dir1[, dir2 ...]* if they are empty
rm -r *dir1 [dir2 ...]*	remove directory *dir1 [dir2 ...]* and subdirectories recursively
mv *dir1 dir2*	rename directory *dir1* (works only, if *dir2* does not exist and its path is in the same filesystem as *dir1*)

File Archives

tar *[c|x|t]vf [files]*	c: archive *files* in a **.tar**-file x: extract archive from **.tar**-file t: list contents of **.tar**-file
cpio *[options] [files]*	archive (parts of) a file system recursively

Metacharacters: Shell Expansion of File Names

The shell is able to interpret metacharacters within filenames on input (wild cards (example): i*t expands to "input")

*****	stands for zero or more arbitrary characters
?	stands for one arbitrary character
[ccc]	single characters from the set ccc, ranges of characters are permitted (Examples: [12r], [j-l])
Suppress expansion	a leading "\" prevents a metacharacter from interpretation by the shell.

Shell Expansion of Commands and Variables

‘cmd‘	‘backquote‘: the whole expression is replaced by the output of **cmd**
$VAR	is substituted by the value of **VAR**
"string"	*‘cmd‘* or *$VAR* within **string** are substituted, no expansion of metacharacters
’string’	no substitution of *‘cmd‘* or *$VAR* within **string** , no expansion of metacharacters

I/O Redirection

Normally the shell expects input from a terminal, and output is also sent to a terminal. Redirection is used to write command output to files or read input from files. UNIX defines three I/O units with corresponding *file descriptors*:

- 0: stdin (standard input)
- 1: stdout (standard output)
- 2: stderr (standard error)

prog > file	redirect (write) stdout of **prog** to **file**
prog >> file	append stdout of **prog** to **file**
prog < file	read stdin for **prog** from **file**
prog <file1 >file2	read stdin for **prog** from **file1**, redirect stdout to **file2**
prog <<EOF **...** **EOF**	here-document: use the following text as stdin for **prog**. **EOF** on a line by itself indicates the end of input
prog 2>file	write stderr of **prog** to **file**
prog 2>&1	with file descriptor: write stderr of **prog** to stdout
set -o noclobber	prevents data from accidentally being overwritten while redirecting output with >

Access Control

Ownership:

u (user)	owner of file
g (group)	group of users (AIX: the active group while creating a file (enter **id** to find out). If the user belongs to several groups, the current group can be changed with the **newgrp** command)
o (others)	all the other users of a system

Controlling file access (independently for owner, group and others):

r:	Files: read permission. Directories: permission to list contents
w:	Files: permission to change contents. Directories: permission to add and delete files
x:	Files: execute permission. Directories: permission to operate on the contents
t:	'sticky bit' (directories): prevents files from being deleted by anyone other than the owner
s:	'setuid-bit' (files): execute a program using the owner's permissions (rather than those of the one who calls it)

chmod *mode file(s)* change the permissions of *file(s)* according to *mode*:

mode may be an **octal number**:

Example: read, write, execute for the owner, read and execute for the group and read for others:

`rwx r-x r--` → 111 101 100 → **754**

mode may be a comma separated list of **permission changes**:

(*chmod g-x,o+r file* no execute permission for group, add read permission for others)

umask *ooo*	Define permissions for new files by an octal number *ooo*, specifying the permissions of the standard permission 666 to be denied. (Suggestion: **umask 077** or **umask 027**)

Shell-Variables (Korn shell)

Assign a value: **NAME=value**

Retrieve the value: **$NAME**

Within Korn shell variables must be exported before they can be used within subshells.

Some Examples of Shell Environment Variables:

HOME	Home directory
PATH	Search path for commands
PS1	Prompt string
USER	Login name of a user
TERM	Terminal type
DISPLAY	X11–Server-Display

Commands:

export *NAME*	Export variable *NAME*
export *NAME=value*	Assign variable *NAME* a value and export it
env	List exported variables

Print Files

lpr -P*printer file*	Print *file* on *printer* (printer name dependent on local configuration)
lpq -P*printer*	Examine printer status
lprm -P*printer jid*	remove print job with job-id *jid*. *jid* can be examined by **lpq**

Command Execution

RETURN (=<ctrl> m)	Execute Command
<ctrl> c	Stop execution of a command
<ctrl> d	End of typed input (End of File Key)
<ctrl> s	Stop terminal output
<ctrl> q	Start terminal output
<ctrl> z	Suspend execution of a command
cmd **&**	Execute *cmd* in the background
bg, **fg**	Reactivate a suspended command in the background (bg) or in the foreground (fg)
nohup *cmd [&]*	'no hangup': execution of *cmd* will continue even if the user logs off the system (exit)
cmd1 **|** *cmd2*	Pipeline: link commands in a way that the standard output of *cmd1* becomes standard input of *cmd2*. *cmd2* is the father of *cmd1*

cmd1 **&&** *cmd2*	*cmd2* is executed only if the execution of *cmd1* ends up successfully
cmd1 **||** *cmd2*	*cmd2* is executed only if the execution of *cmd1* does not end up successfully
cmd1 **;** *cmd2*	execute *cmd2* after execution of *cmd1* stopped

Filter Commands

grep *pattern [files]*	Search for *pattern* within standard input or *files*, if specified. *pattern* can be a regular expression including the following meta characters: \ prevent from interpretation as a meta character **.** arbitrary, single character **[...]** any character from [...] **r*** repitition of character r **^,$** beginning of line (^), end of line (**$**)
cut *-f|-c file*	extract characters or fields from lines
sort *[key]... [file]...*	sort lines from *file* according to *key*. Read from stdin if - is specified instead of *file*
tr *str1 str2*	replace *str1* by *str2*
awk, sed	programming languages for data manipulation (**awk**: C-like, **sed**: vi-like).

X-Windows

An X-Windows-Server is a process that creates a window on the user's desktop-system display. An X-Windows-Client is an application process that is responsible for a window's contents.

[open] xinit ; exit	start X-Windows and terminate the console session
xhost *c-host*	server command: permit *c-host* to open a window on the server's display
DISPLAY=*s-host***:0.0**	client command: instruct the client
export DISPLAY	process to open a window under *s-host*'s window manager
eval 'resize'	running a network terminal emulation: solve resizing problems after changing an *xterm*'s window size

Attention !!! To log off a server system from within an X-Windows-Session it is **not** sufficient to close all windows. Instead, the window manager itself has to be stopped.