

Introduction to Unix

Klaus Macherey

**Human Language Technology and Pattern Recognition
Lehrstuhl für Informatik VI
Computer Science Department
RWTH Aachen University, Germany**

Contents

1. Introduction

- Linux file system hierarchy

2. Basic Commands

- cd, mv, ls, ln, mkdir, rm, ...
- Online Man-Pages

3. Login Shells

- bash, tcsh
- Setting environments and aliases

4. Data Manipulation Using Filters

- Pipes, Redirection
- Filters like grep, sort, gawk, sed

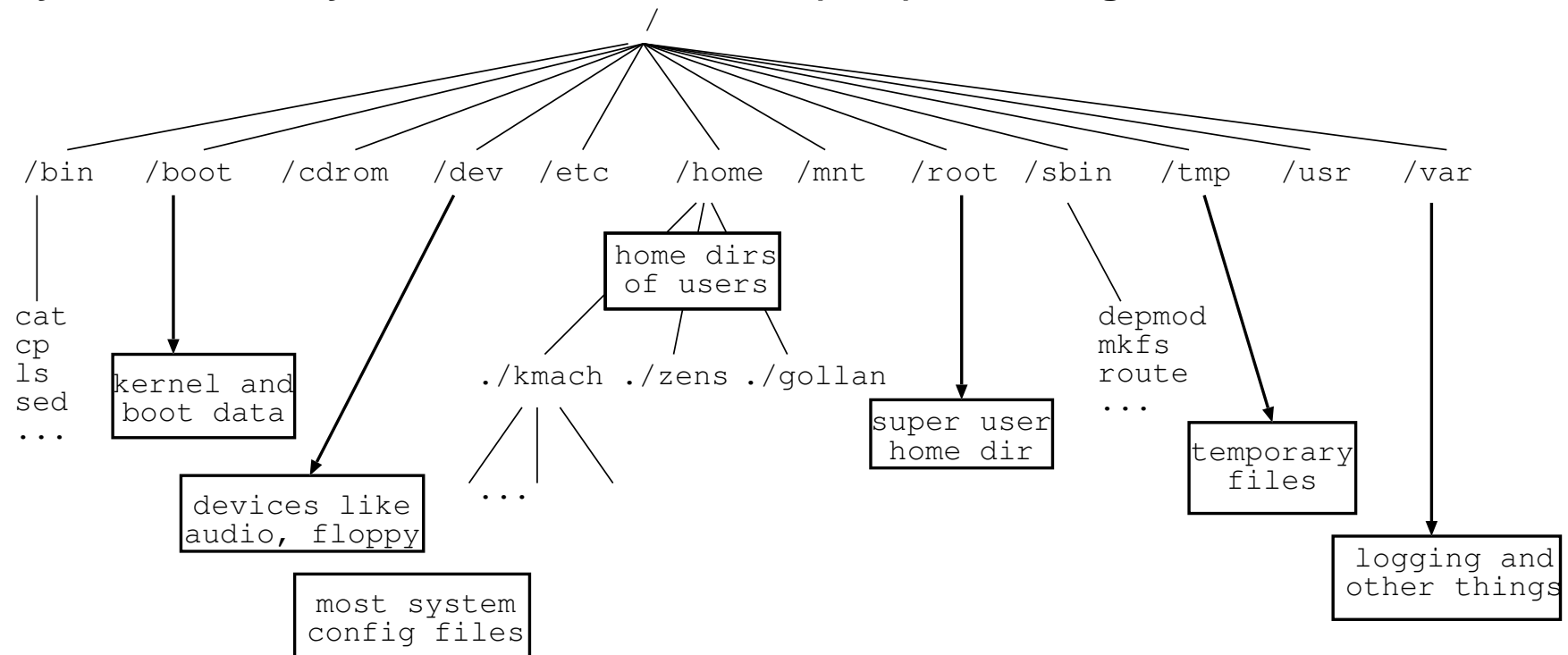
5. Job Control

- Starting Processes
- Process Snapshots using top
- Daemons and Zombies

6. Remote Login

Introduction

Although Linux provides a graphical user interface, commands are usually invoked from a **shell**. The file system hierarchy of Linux starts with **/** (root) and is organized as follows.



The superuser (administrator) is also called root. After logging into to the system, the user can type in commands in his default shell (`echo $SHELL` prints the name of the current shell).

Additionally, graphical user interfaces can be used like

- kde
- gnome
- ...

Basic Commands

list directory contents:

```
ls /
bin  cdrom  etc      floppy  man     misc    net     proc    sbin    var
boot dev     home     lib     mnt     opt     root    usr     tmp
```

```
ls -l /bin
-rwxr-xr-x 1 root root 31724 2004-07-16 13:37 df
-rwxr-xr-x 1 root root 13912 2004-07-16 13:37 echo
-rwxr-xr-x 1 root root 12044 2004-04-13 07:26 kill
-rwxr-xr-x 1 root root 55340 2004-07-16 13:37 mv
```

...

permissions: user : group : others (r)ead, (w)rite, e(x)ecute, ...

Each file and directory has as set of permissive flags: e.g. whether a file is executable is a property of a file and not determined by its name.

change directory:

```
cd /home/kmach
cd
cd ~
```

all commands change to the home directory of the user (here: kmach)

Basic Commands (cont'd)

creating directories/files:

```
mkdir      /home/kmach/bin  
mkdir -p   /home/kmach/src/syspro/uebung01
```

create directories in home directory (-p = parent)

```
emacs /home/kmach/src/syspro/uebung01/aufgabe1.c
```

invoke editor and edit file aufgabe1.c deleting directories/files:

```
rm /home/kmach/src/syspro/uebung01/aufgabe1.c  
rmdir /home/kmach/src/syspro/uebung01  
rm -rf /home/kmach/src
```

Last command deletes COMPLETE src directory (!)

Online Manual Pages

```
man rm  
man man  
man -k delete
```

Option -k searches for keywords (prints all manual pages containing this keyword)

Manual Pages

Manual pages are organized in different sections:

- 1 Executable programs or shell commands
- 2 System calls (functions provided by the kernel)
- 3 Library calls (functions within program libraries)
- 4 Special files (usually found in /dev)
- 5 File formats and conventions eg /etc/passwd
- 6 Games
- 7 Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
- 8 System administration commands (usually only for root)
- 9 Kernel routines [Non standard]

Some manual pages have the same name. Use the section info in order to get the correct one:

man 1 free (displays amount of free and used memory)

man 3 free (description of standard library function free)

i-Nodes and Links

i-nodes

Linux file systems keep the name of a file (filename part) and its content (data part) separate within so-called **i-nodes**. Information about files like their names, permission flags, etc are stored in these i-nodes. An i-node only contains pointers to data-blocks but not the data itself. This has the advantage that moving a file to another directory only means that the filename must be moved and not its content.

links

Links are special files that 'point' to already existing files. They only consist of a link name and an already existing i-node number.

```
ln -s /home/kmach/src/syspro ~/syspro          create symbolic link to  
                                              syspro directory
```

It is possible to create cyclic link structures which in the past confused commands like ls.

Login Shells

Usually, bash or tcsh are used as default shells. User defined configurations as follows

bash

used as login shell: `~/.bash_profile`, `~/.bash_login`, and `~/.profile`

used as interactive shell: `~/.bashrc`

use export to set environment variables

```
cat ~/.bashrc
export EDITOR='which emacs'
export PATH=./:${HOME}/bin:${PATH}
test -s ~/.alias && . ~/.alias
```

Read and execute commands from the filename argument in the current shell context (`.` synonym for `source`)

```
cat ~/.alias
alias ls='ls --color=yes'
alias la='ls -la --color=yes'
alias ll='ls -l --color=yes'
```

tcsh

used as login shell: `~/.tcshrc`, `~/.cshrc`, `~/.history`, `~/.login`

used as interactive shell: `~/.cshrc`

use setenv to set environment variables

```
cat ~/.cshrc
setenv EDITOR /usr/bin/emacs
```

Pipes and Redirection

Output of prog1 can be used as input of prog2 using pipes |

```
echo "hello world" | wc
      1      2     12
cat 80days.txt | grep "The" | head -n2
```

The way in which he got admission to this exclusive club
The game was in his eyes a contest, a struggle with a difficulty,

Output can be redirected into files using >

```
cat 80days.txt | grep "The" | head -n2 > ~/two_lines.txt
```

Unix provides different channels: 0 input, 1 output, 2 error

```
cat /hallo
cat: /hallo: No such file or directory
```

```
cat /hallo 2> ~/error_message (BASH only)
```

Redirect stdout and stderr using 2>&1

Using Filters

AWK is a line-oriented language.

```
echo "what a wonderful world" |\  
gawk 'BEGIN{  
    printf("#####\n");  
}  
{  
    for(i=1; i<=NF; ++i) printf("%s\n", $i);  
}  
END{  
    printf("#####\n");  
}'
```

```
#####  
what  
a  
wonderful  
world  
#####
```

Using Filters (cont'd)

awk scripts process input files line-wise, i.e. each line of the input file is processed according to the commands specified in the middle block.

Simple awk scripts are constructed as follows: the optional `BEGIN{ }` block contains commands that are executed only once *before* the first line is processed. The optional `END{ }` block is processed after the *last* line has been processed by the middle block.

Each line is separated into a number of entries which can be referenced by `$1`, `...`, `$NF`. For `for` and `while` loops can be used with a syntax similar to c-programs. `$0` refers to the entire line. `$NF` is a special variable and set for each line referring to the number of fields.

Details can be found reading the manual page

`man gawk`

Using Filters (cont'd)

sed stream editor.

```
echo "SysPro 2004" | sed s/'2004'/'2005'/g
```

```
SysPro 2005
```

sort sorting texts

```
echo -e " eins \n zwei \n drei \n vier" | sort
```

```
drei
```

```
eins
```

```
vier
```

```
zwei
```

```
echo -e " 10 \n 9 \n 8 \n 7 " | sort -n
```

```
7
```

```
8
```

```
9
```

```
10
```

Jobs and Processes

Linux is a multiuser system. Therefore, processes are always associated with a user id. A user can only start/kill jobs that have the appropriate user id.

To start a job, use `chmod a+x [FILENAME]` to set the executable flag and type `./[FILENAME]` in your shell. Jobs can be cancelled pressing `<CNTRL>+c`. Jobs can be executed in foreground and in background mode. Type `./[FILENAME] &` to start the job in background mode.

`kill` is used to send different signals to a running process.

E.g. `kill -9 <ProcessNumber>` terminates a job with the given process number.

The process number can be identified using `top` or `ps aux | grep [JOBNAME]`

Process Control

Starting jobs::

use `chmod a+x` **in order to activate execution flags**

start job by typing in jobname

key commands:

`<cntrl>+c` **cancels job**

`<cntrl>+z` **suspends job**

type `fg` **after suspension in order to continue job in foreground**

type `bg` **after suspension in order to continue job in background**

yes

Y

Y

Y

Y

`<cntrl z>`

`bg`

Y

Y

Y

Y

Process Control (cont'd)

top

Tasks: 152 total, 4 running, 148 sleeping, 0 stopped, 0 zombie
 Cpu(s): 94.9% us, 4.0% sy, 0.0% ni, 0.0% id, 0.0% wa, 0.5% hi, 0.7% si
 Mem: 1034416k total, 1009024k used, 25392k free, 10384k buffers
 Swap: 3906464k total, 105384k used, 3801080k free, 332660k cached

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
12946	kmach	25	0	124m	113m	4432	R	79.9	11.2	0:08.02	cclplus
12952	kmach	20	0	60712	47m	4372	R	69.3	4.7	0:02.09	cclplus
12945	kmach	25	0	119m	109m	4372	R	48.1	10.8	0:05.79	cclplus
3274	kmach	15	0	12656	9752	3876	S	1.0	0.9	0:05.19	emacs
12948	kmach	17	0	1944	988	736	R	0.7	0.1	0:00.14	top
572	sgeadmin	16	0	4144	2240	1804	S	0.3	0.2	54:28.57	sge_execd
12017	kmach	16	0	31352	13m	10m	S	0.3	1.4	0:02.24	konsole
1	root	17	0	1584	448	424	S	0.0	0.0	0:01.29	init
2	root	RT	0	0	0	0	S	0.0	0.0	0:00.12	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/0
4	root	RT	0	0	0	0	S	0.0	0.0	0:00.02	migration/1
5	root	34	19	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/1
6	root	10	-5	0	0	0	S	0.0	0.0	0:07.53	events/0
7	root	10	-5	0	0	0	S	0.0	0.0	0:00.13	events/1
8	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	khelper

Daemons and Zombies

Daemons

Processes running permanently in background mode. Daemons are usually started at boot time and are terminated during system shut down. Daemon processes are responsible for testing the printer queue, checking for incoming email, etc.

Zombies

In Unix/Linux, parent processes should always survive their child processes. A zombie is a "child" process that was started by a "parent" process but then abandoned by the parent. A process that has died but has not yet relinquished its process table slot (because the parent process has not executed a wait for it yet) is called a zombie.

Remote Login

In order to login to different machines within a network, use ssh:

```
kmach@rhodium: ssh kupfer
kmach@kupfer: ls -l ...
```

You can use scp in order to copy files from a remote machine to your local machine using a secured channel:

```
kmach@rhodium: scp kmach@kupfer:/tmp/myfile.c /home/kmach/myfile.c
```

This copies a file from kupfer to your local machine (here rhodium)

Literature & Online Tutorials

<http://www.ee.surrey.ac.uk/Teaching/Unix/>

<http://www.isu.edu/departments/comcom/unix/workshop/unixindex.html>

<http://www2.ocean.washington.edu/unix.tutorial.html>

Unix for the Impatient, P. W. Abrahams, B. A. Larson, Addison Wesley

Linux/Unix Systemprogrammierung, Hemit Herold, Addison Wesley