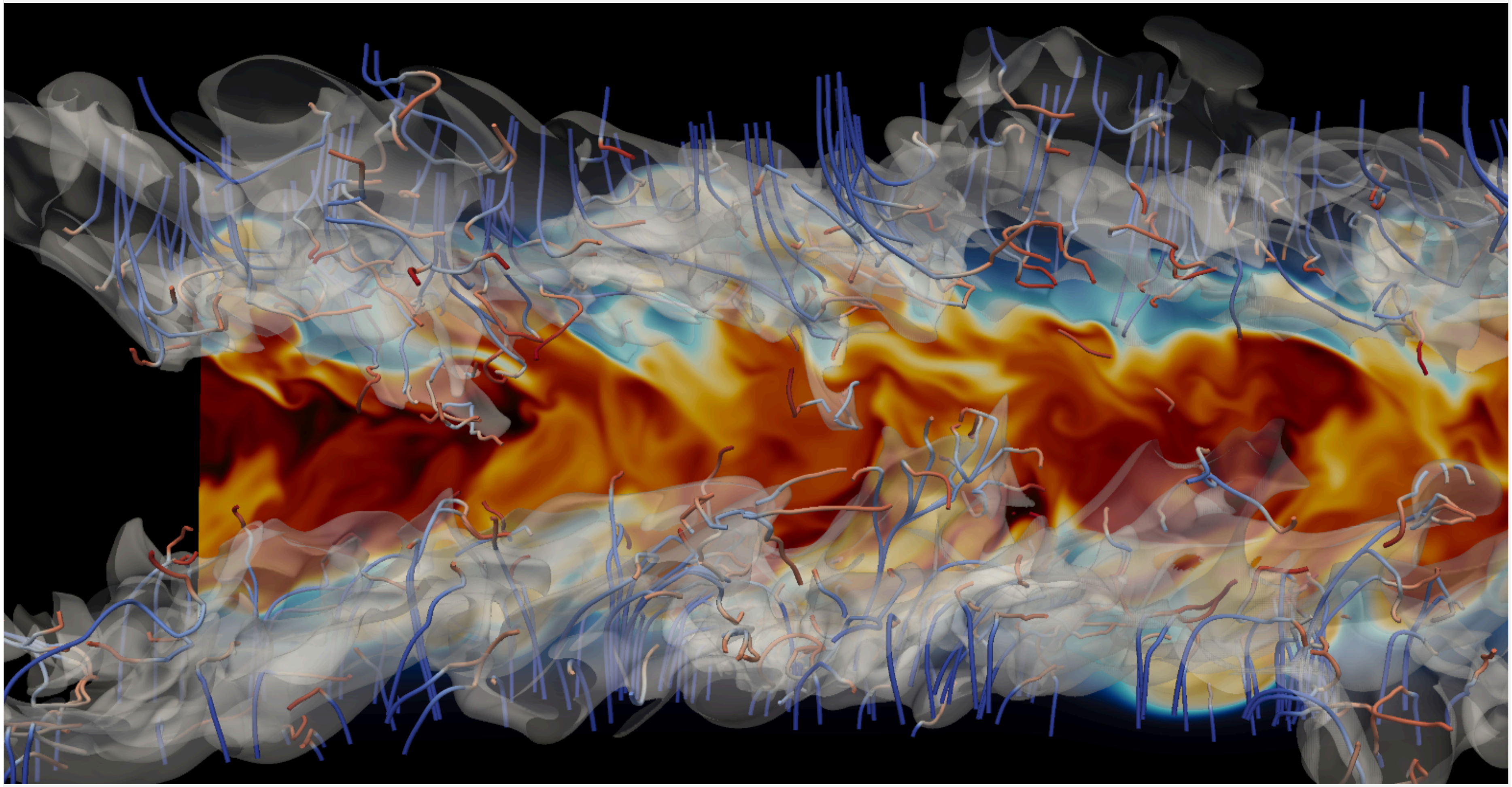
Software Tools for UNIX/Linux Systems

Part 2: Basic

C. Hasse

TECHNISCHE UNIVERSITÄT DARMSTADT



Agenda

Shell – what is a shell

- classic user interface on UNIX
- command line interpreter
- "shell" around the unix "kernel"
- basic text input
- shell is just a program

http://farm3.staticflickr.com/2241/2394105211_6e1fb55a30_z.jpg?zz=1

Shell - history

- osh shell 1971-79 (Ken Thompson) [Research Unix]
- sh - bourne shell 1978 (Stephen Bourne) [Unix v7]
- csh/tcsh - C shell 1978 (Bill Joy) [BSD]
  - c - like syntax
- ksh – Korn Shell 1983 (David Korn) Bell Labs
  - compatible with sh but copied features from csh
- bash - Bourne Again Shell1989 (Brian Fox) [cross plattform]
  - open source clone of sh
  - default on many linux distributions
  - we will use this one during the course
- zsh - Z-shell 1990 (Paul Falstad) [cross plattform]
  - a students pick of bash, ksh and tcsh features
  - popular with developers

Shell - usage

- start terminal emulator (Terminal)
- default shell is started
- lets start another to show that its just a program
  - ps aux | grep username
- logout (End of Transmission Character CTRL+D)
  - (EOT)
  - exit

Login session

- determine your credentials
  - finger student
  - who am i
  - whoami
  - id student
  - who
  - last

Commands and arguments

- commands are usually followed by a number of arguments which are space separated

```
$> date
Fri Sep 23 10:12:45 CEST 2016
$> touch a b c d
$> ls
a  b  c  d
$> wc /etc/passwd
42   67 2075 /etc/passwd
$> wc -l /etc/passwd
42 /etc/passwd
```

Path specification and autocompletion

TECHNISCHE UNIVERSITÄT DARMSTADT

- unix directories are build like tree structures starting with "/" the so called root of the fileystem

path	description
/	filesystem root
/boot	files needed to boot
/bin	fundamental binaries used by all users
/dev	file representations of devices, pseudodevices
/etc	system wide configuration
/home	user home directories
/lib	system libraries
/usr	executables,libraries, resources which are not system critical
/var	place for files that may change often

Path specification and autocompletion

- folder structure differs on Linux/Unix implementations
- Refer to Filesystem Hierarchy Standard for reference on Linux systems

```
$> cd /
$> ls
$> cd etc
$> cd p(Tab)(Tab)
```

– reference to current folder and the one above in every folder "." and ".."

```
$> cd .
$> cd ..
```

Path specification and autocompletion

- special paths
  - $> cd ~ go to home directory
  - $> cd – go to last directory

- autocompletion is triggered using TAB
- completions for bash are stored in /etc/bash_completion.d

$> cd /e(tab)/pam(tab) (tab)(tab)

notice how files are not completed while in cd command

Environment Variables

Variable	Description
USER	current username
PATH	application search path separated by colon
DISPLAY	name of X11 display to use
SHELL	name of the current shell
TERM	terminal name (used to determine terminal capabilities)
TERMCAP	stores termcap db or terminal escape sequence (see above)
OSTYPE	type of operating system
MACHTYPE	describes cpu architecture
EDITOR	user preferred editor
PAGER	user preferred text pager (text viewer)
MANPATH	search path for documentation (man pages)

Environment Variables

```
$> echo $PWD
$> pwd
$> which cmake
/usr/bin/cmake
$> export PATH=/opt/bin:$PATH
$> echo $PATH
$> which cmake
/opt/bin/cmake

$> env
```

Command history

```
$> history [num]
(display entire history or last num entries)

$> up/down (enter to execute again)

$> (Ctrl+R) ma (reverse incremental search)

$> !whi (execute last line starting with same
phrase -> should be „which cmake“)
```

- preceed space → command not included in history

Text redirection

- the shell can combine multiple commands in various ways
- text is the glue code

This is the Unix philosophy:

"Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface."

- Douglas McIlroy (inventor of pipes, spell, diff, sort, join, speak, tr)

Text redirection

```
$> ls -l | wc –l
$> ls -l > testfile
$> more testfile
$> ls -l >> testfile
$> more testfile
$> wc -l < testfile
$> wc -l < testfile > test2
$> wc -l > text << EOF
heredoc> bla
heredoc> this
heredoc> EOF
$> more text
        2
```

Text redirection

- special filestreams
  - 0 stdin standard input
  - 1 stdout standard output
  - 2 stderr standard error

```
$> ls /bla > out (/bla shouldn't exist)
$> ls /bla > out 2> out.err

$> more out
$> more out.err
```

Job control

- shell can run jobs (programs) in the background and has job control features

```
$> ps
$> sleep 50& (ampersand backgrounds the task)
$> ps
$> fg
$> Ctrl+C
$> ps
```

- jobs however are terminated when shell closes (hangs up)

```
$> nohup okular Desktop/SWP/announcement.pdf &
$> okular Desktop/SWP/announcement.pdf &
```

User managment

- user accounts have some basic fields

field	description
login	login name
password	password
full name	full user name
uid	unique number repesents user
gid	unique number represents primary group membership
home-directory	e.g. /home/user
user shell	e.g. /bin/bash
account-expire-date	e.g. 30.09.2013
password-expire-date	e.g. 30.06.2013

SSH

- network protocol to encapsulate services in cryptographic wrapper
  - remote shell connection
    - ssh kurs51xxx@lcluster1.hrz.tu-darmstadt.de
    - lsb_release -a
  - remote file copy
    - scp test.dat user@hostname:~
  - network connections
    - ssh -L80:127.0.0.1:80 user@hostname
  - filesystem access sshfs
  - - mkdir hostname_home
    - sshfs user@hostname:~ hostname_home

File permissions

- also apply for folders and devices (with some restrictions)

bit value	permission	listing
0	no read, no write, no execute	---
1	no read, no write, execute	--x
2	no read, write, no execute	-w-
3	no read, write, execute	-wx
4	read, no write, no execute	r--
5	read, no write, execute	r-x
6	read, write, no execute	rw-
7	read, write, execute	rwx

File permissions

- permission bits are masked in three fields controlling access for
  - user: exact uid
  - (primary) group: exact gid
  - other (every other uid/gid)

- there is also the first field indicating different kinds of files → see table

Symbol	Meaning
-	regular file
d	directory
l	link
c	character device
s	socket
p	named pipe
b	block device

File permissions

```
$> ls -la
drwxr-xr– 2 user group 2 13 Jun 18:45 test
```

- "test" is a folder
- user „user“ may read, write, execute
- group „group“ may read and execute
- everyone else may read
- it contains 2 elements (. and ..)
- was created 13. June 18:45

File permissions

chmod command allows to change permissions

absolute notation e.g. (recursive)

```
$> chmod -R 755 test
```

or using symbolic permissions which allow relative and absolute change

option	letter	represents
(who)	u	User
(who)	g	Group owner
(who)	o	Other
(who)	a	All (“world”)
(action)	+	Adding permissions
(action)	-	Removing permissions
(action)	=	Explicitly set permissions
(permissions)	r	Read
(permissions)	w	Write
(permissions)	x	Execute
(permissions)	t	Sticky bit
(permissions)	s	Set UID or GID

File permissions

```
$> touch test
$> chmod 666 test
-rw-rw-rw-   1 user group 0 13 Jun 19:04 test
$> chmod go-w,a+x test
-rwxr-xr-x   1 user group 0 13 Jun 19:04 test
```

- remove write permission for group and others
- add execute permissions to all

File permissions

- sticky bit (1) only common on folders today (used to be whether a program was kept in fast memory)
- chmod 777 of top folder
  - everybody could delete/rename everything
- chmod 1777 of top folder
  - only owner, group and root may rename/delete files/folders

```
drwxrwxrwt 2 user group 2 13 Jun 19:16 bla
```

setuid/setgid (4)/(2) make programs run with owner (user/group) permissions of the file when applied on executables (not with permissons of the user who started the program)

setgid applied on a top folder forces new files and folders to inherit the group id of the top folder (setuid does nothing on directories by default)

File permissions

- chown commands allows to set owner/group of files

```
$> chown –R user:group test (recursively)
$> chown –R user: test (recursively, default group of user)
```

- some Unix systems support extended permission flags
  - chflags command
  - $> ls –lo #to display
  - please refer to man chflags(1)

Tools - filesystem tools

- df show free space available on system
- du [path] disk usage of specified path summing with argument -s
- mount (umount) filesystems from any devices may be mounted into each other (always a folder)
- e.g. usb pen drive in /media/mydisk

- warning: filesystems like FAT-32 and NTFS do not support all permissions available in Linux/Unix

Tools - file operation tools

- cp copy files (-r recursively)
- mv move files
- ln create hard links - only one filesystem (filesystem feature: blocks are linked)
- ln -s create soft links - allows linking between filesystems
- mkdir create folder (-p create all parents too)
- touch updates time stamp or creates empty file if it doesn't exist
- file determine file type file /bin/bash
- ls list directory contents

Tools - file operation tools

- scp secure copy across machines
- $> scp file user@hostname:~
- rsync (may use ssh) allows syncing of file/directory states
- $> rsync -avz files user@hostname:~
  - archive, verbose, compress
  - checks for existing files, won't overwrite what is up to date
- compression: tar, zip
  - compress $> tar czf files.tar.gz files
  - decompress $> tar xf files.tar.gz

Tools - text related tools

- pagers/display tools
  - more, less: classic pagers
  - head, tail: display files from top or bottom
- calculators: bc, dc
- editing tools
  - cut : cut out selected portion of each line
  - paste: merge several files linewise
  - tr : "translate" characters from inputstream by replacing them with specified substitutions/deletions to outputstream
  - more advanced tools like sed, awk, ... will be discussed later

Getting help

- man pages - universal manual pages
- man man to get started
  - text is displayed in pager
    - scroll screen up/down (Ctrl+f, Ctrl+b)
    - linewise up/down j/k up/down
    - search forward /
    - search backward ?
    - help h (? on some systems)
- apropos, whatis search for keywords in man database
- whereis to find command or man page
- info more thorough documentation e.g. info bash

Getting help

man categories and their respective introduction page

category	description
intro(1)	introduction to general commands (tools and utilities)
intro(2)	introduction to system calls and error numbers
intro(3)	introduction to the C libraries
intro(4)	introduction to special files
intro(5)	introduction to file formats
intro(6)	introduction to games
intro(7)	miscellaneous information pages
intro(8)	introduction to system maintenance and operation commands
intro(9)	introduction to system kernel interfaces

Getting help

- RTFM

- http://tldp.org The Linux Documentation Project
- http://debiananwenderhandbuch.de
- http://debian-handbook.info/browse/stable/
- http://www.freebsd.org/doc/de/books/handbook/FreeBSD Handbook (very well written)

- Selection by the authors:

- https://www.tecmint.com/use-wildcards-to-match-filenames-in-linux/
- https://www.linux.com/blog/2018/8/linux-beginners-moving-things-around
- https://www.tecmint.com/understanding-shell-initialization-files-and-user-profiles-linux/
- https://www.tecmint.com/customize-bash-colors-terminal-prompt-linux/
- https://www.tecmint.com/difference-between-su-and-su-commands-in-linux/
- https://www.tecmint.com/linux-command-line-tricks-and-tips-worth-knowing/

Exam dates

- **FIRST**: sign in to moodle survey https://moodle.tu-darmstadt.de/mod/scheduler/view.php?id=448606
- **THEN**: wait until we tell you that TuCan sign in is possible
- **FINALLY**: sign in for the exam in TuCan