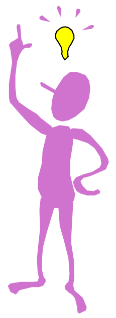
Introduction to UNIX

BUPT/QMUL

2014-2-27



Contents

continued

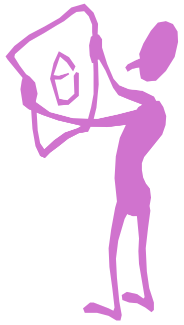
Contents

1. Background on UNIX

1.1. What is UNIX?

1.2. History

1.3. Why use UNIX?



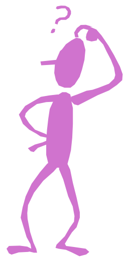
1.1. What is UNIX?

- The UNIX Operating System (OS) is a large program (mostly coded in C) that turns the computer into a useable machine.
- It provides a number of facilities:
  - management of hardware resources
  - directory and file system
  - loading / execution / suspension of programs

1.2. Brief History

• 1969	First UNIX at Bell Labs
• 1975	Bell Labs makes UNIX freeware
• 1970's	Berkeley UNIX (BSD)
• 1980's	TCP/IP MIT X-Windows
• 1990's	The Web, LINUX (e.g. RedHat 9.0, 红旗 5.0)

1.3. Why Use UNIX?



- multi-tasking / multi-user
- lots of software
- networking capability
- graphical (with command line)
- easy to program
- portable (PCs, mainframes, super-computers)

continued

- free! (LINUX, FreeBSD)
- popular
- not tied to one company
- active community

2. Starting / Finishing

2.1. Your Account

2.2. Login to your Account

2.3. Password Tips

2.4. Logout from your Account

2.1. Your Account

- Each user has their own space, called their *account*.
- Type your login ID and password to enter your account.
- Only if the login ID and password match will you be let in.

2.2. Login to your Account

`login: ad`	You type your ID and RETURN.
`Password:`	You type your password and RETURN. It does not appear.
`$`	The UNIX prompt (or similar). You can now enter commands.
`Access denied` `Password:`	Login ID and password not match

2.3. Password Tips

- NEVER tell anyone your password.
- Don’t write it down.
- A good password is:
  - 8 (or more) characters long
  - uses a mix of uppercase and lowercase letters, numbers, and symbols (e.g. #, %).
- You can change your password with the `passwd` command (see later).

2.4. Logout from your Account

```
logout
```

or

`^D`	Press CONTROL and D together

or

```
exit
```

13

3. Typing UNIX Commands

3.1. The Shell

3.2. Typing Commands

3.3. Control Characters

3.4. Changing your Password

14

3.1. The Shell

- The UNIX user interface is called the *shell*.
- The shell does 4 jobs repeatedly:

$$\begin{tikzcd} & \text{display prompt} \arrow[dl, bend right] & \\ \text{read command} \arrow[dr, bend right] & \textit{the shell} & \text{execute command} \arrow[ul, bend right] \\ & \text{process command} \arrow[ur, bend right] & \end{tikzcd}$$

15

3.2. Typing Commands

- Try these:

```
date
cal 3 1997
who
ls
ifconfig
man cal
clear
```

`man cal`	Press enter for next line; Press spacebar for next page; ^C or q to stop

16

3.3. Control Characters

- Erasing characters

`DELETE`	delete last character
`^H`	delete last character (press CONTROL and H together).
`^W`	delete last word
`^U`	delete the line

- Very useful control characters

`^C`	terminate command
`^S`	suspend output
`^Q`	resume output

Special Characters can be Altered

- Show current settings:
```
stty -a
```

- Change a setting:
```
stty erase ^?
```
Type `^` and `?`.

- Reset:
```
stty sane
```

3.4. Changing your Password

- The command is:
```
passwd
```

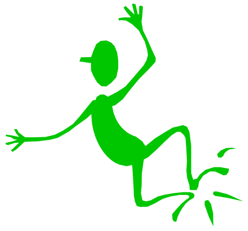
- It will ask you for the new password twice.

4. Command to Use Right Away

4.1. Date Commands

4.2. You and the System

4.3. Calculators



4.1. Date Commands

• **`date`**	Gives time and date
• **`cal`**	Calendar

```
cal
cal 1997
cal 3
cal 7 1962
cal 9 1752
```

Not a mistake. Why?

4.2. You and the System

• **`uptime`**	Machine’s ‘up’ time
• **`hostname`**	Name of the machine
• **`whoami`**	Your account name

4.3. Calculators

• `xcalc`	Requires X-Windows
• **`expr`** `e`	Simple arithmetic
• **`bc`**	Programmable Calculator

```
expr 3 + 5 + 7
```

Using bc

- bc

```
3 + 5 + 7
   17          Output
^D
```

- bc -l   Use Maths library

```
scale=3        Set display to 3 dp
150/60
l(30)          natural log function
^D
```

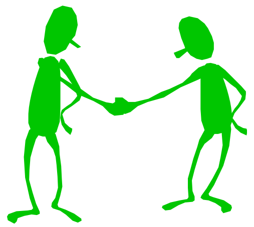
- bc

```
obase=2        Output base
ibase=16       Input base
FFC1
^D
```

5. UNIX Help

5.1. On-line Help

5.2. UNIX books



5.1. On-line Help

- **man**   Manual pages

```
man cal
man man
```

- apropos *topic*   Lists commands related to topic

```
apropos game
apropos help
```

`man -k topic`	Same as `apropos`
`whatis cmd` `whatis find`	One-line description
`where cmd`	Location of command
`which cmd`	Location

- **`locate`** `cmd` — List files with `cmd` in their name (or path)

```
locate game
```

- Output of these commands can be very long. See one screenful at a time with: `| more`

```
locate game | more
apropos print | more
```

- Press enter/spacebar to go on; `^C/q` to stop.

5.3. UNIX Books

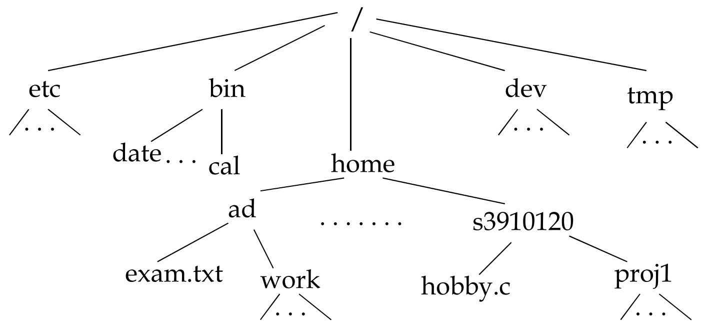
- *A Student's Guide to UNIX*, Harley Hahn, McGraw-Hill, 1993.
- *A Practical Guide to the UNIX System*, Mark G. Sobell, Benjamin-Cummings, 3rd Edition, 1995.
- *An Introduction to Berkeley UNIX*, Paul Wang, Wadsworth, 1992.
- … …

6. The UNIX File System

6.1. An upside-down Tree
6.2. Some System Directories
6.3. Where do you login?
6.4. Pathnames
6.5. Commands and Pathnames

6.1. An upside-down Tree

- A simplified UNIX directory/file system:



33

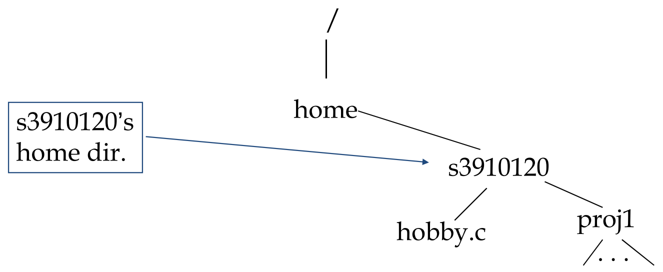
6.2. Some System Directories

• `/`	*root* directory
• `/bin`	commands
• `/etc`	system data files (e.g. `/etc/passwd`)
• `/dev`	files representing I/O devices

34

6.3. Where do you login?

- Your *home directory*, which is named after your login ID.



35

6.4. Pathnames

- A *pathname* is a sequence of directory names (separated by /'s) which identifies the location of a directory.
- There are two sorts of pathnames
  - absolute pathnames
  - relative pathname

36

Absolute Pathnames

- The sequence of directory names between the top of the tree (the *root*) and the directory of interest.
- For example:

```
/bin
/etc/terminfo
/export/user/home/ad
/export/user/home/s3910120/proj1
```

Relative Pathnames

- The sequence of directory names below the directory where you are now to the directory of interest.
- If you are interested in the directory `proj1`:

`proj1`	if you are in `s3910120`
`s3910120/proj1`	if you are in home

6.5. Commands and Pathnames

- Commands often use pathnames.
- For example:

`cat /etc/passwd`	List the password file

7. Working with Directories

7.1. Moving between Directories

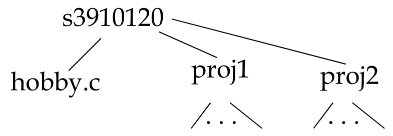
7.2. Special Directory Names

7.3. Investigate the System

7.4. Making / Deleting / Renaming Directories

7.1. Moving between Directories

- s3910120's home directory:



41

- If you are in directory `s3910120` how do you move to directory `proj1`?

```
cd proj1
```

- You are now in `proj1`. This is called the *current working directory*.

42

- **`pwd`** Print name of current working directory

- Move back to directory `s3910120` (the parent directory):

```
cd ..
```

43

- When in `proj1`, move to `proj2` with one command:

```
cd ../proj2
```

- `../proj2` is a *relative* pathname

44

7.2. Special Directory Names

/	The root directory
.	The current working directory
..	The parent directory (of your current directory)
~	Your home directory
`~user`	Home directory of `user`

Examples

`cd /`	Change to root directory
`cd ~`	Change to home directory
`cd`	(Special case; means `cd ~`)
`cd ~ad`	Change to ad's home dir.
`cd ../..`	Go up two levels.

7.3. Investigate the System

- Use `cd`
- **`cat`** `file` List `file`

```
cd /etc
cat passwd
```

- **`ls`** Directory listing

```
ls                  List current dir.
ls /etc             List /etc
ls -F               -F option shows types
        a.tcl*      ns-allinone-2.29/
```

7.4. Making / Deleting / Renaming Directories

- Usually, you can only create directories (or delete or rename them) in your home directory or directories below it.

`mkdir`	Make a directory
`rmdir`	Delete a directory
`mv`	Rename a directory

- Create a `lab` directory in your home directory:

```
cd ~
mkdir lab
```

- Create two directories inside the `lab` directory:

```
cd lab
mkdir week1
mkdir week2
```

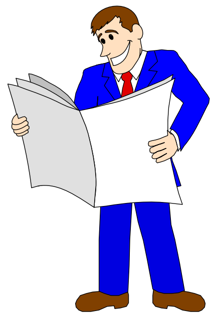
- Delete the `week1` directory:

```
rmdir week1
```

- Change the name of `week2` to `all-weeks`

```
mv week2 all-weeks
```

8. Working with Files



8.1. Creating a Text File

- A quick way:

```
cat > file
```

- This will feed the text you type at the keyboard into `file` until you type `^D` (CONTROL and a D together).

- A more powerful way is to use **`vi`**, a full screen editor (see later).

8.2. Listing Files

• **cat** file cat hobby.c cat /etc/passwd	List the file
• **more** file	List the file a screen at a time. Type spacebar to go on; `^C` to stop

53

• **less** file	Like `more` but more powerful
• **head** file	List the *first* few lines
• **tail** file	List the *last* few lines

54

8.3. Filename Conventions

- Many files have a name and an extension:

`file.c`	A C program
`file.cpp`	A C++ program
`file.txt`	A text file

- However, you can call a file *anything*. It doesn't have to have an extension.

55

8.4. Other Basic Commands

• **cp** file1 file2	Copy `file1`, making `file2`
• **mv** file1 file2	Rename `file1` as `file2`
• **rm** file rm -i file	Delete `file` check first

56

• **wc** `file`	Counts the lines, words, characters in `file`
• **grep** `string file`	Search `file` for `string`

e.g., `grep abc test`

- List lines containing 'Andrew' in `/etc/passwd`
  `grep Andrew /etc/passwd`
- Lines containing 'printf(' in `hobby.c`
  `grep 'printf(' hobby.c`
- Lines starting with 'loca' in `/usr/dict/words`
  `grep ^loca /usr/dict/words`

8.5. Printing

• `lpr file`	Print `file`
• `lpq`	List the print queue. Each print job has a number.
• `lprm job-number`	Remove that print job

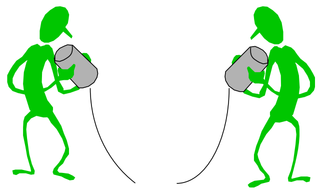
- You may have to name the printer with the `-P` option:
  `lpr -Plj5 hobby.c`
- `lpq` and `lprm` understand the `-P` option

8.6. I/O Redirection

- Most commands output to the screen
  `ls`
- Output can be *redirected* to a file with'>':
```
ls > dir.txt
cal 1997 > year1997
```
- Output can be *appended* to a file with '>>'
```
cal 1997 > years
cal 1998 >> years
```

- Concatenate two files:
```
cat f1 f2 > fs
```
- Input redirection (less common) uses '<'
```
wc < years
```
- Combine input and output redirection:
```
wc < years > year-counts
```

9. Communicating with People



9.1. Information on Others

• **`users`**	Who else is logged on?
• **`who`**	Information on current users
• **`ps`**	What are people doing?

• **`w`**	What are people doing?
`w -sh`	A shorter report

• Examine password info:

```
more /etc/passwd
grep s38 /etc/passwd
```

9.2. Fingering People

• **`finger`**	Info. on current users
`finger -l`	Longer information
• `finger user`	Information on user (need not be logged in)
`finger ad`	

• `finger @machine-name`	User info. for that machine

```
finger @catsix
finger @ratree.psu.ac.th
```

• **`ping`** `machine-name`	Is machine alive (on)?
`ping catsix`	(`^C` to stop)

Your Finger Information

• `chfn`	Change your finger entry

• `finger` also prints the contents of the `.plan` and `.project` files in your home directory. List '.' files with:

```
ls -a
```

9.3. Talking

- `talk user` Talk to user (on any machine)

```
talk ad
talk bill-gates@ratree.psu.ac.th
```

Get out by typing `^C`

Access to remote machines

- Using root account
  sudo -i
- Make sure you have installed SSH server
  ps -ef | grep sshd
  If not, install SSH server
  apt-get install openssh-server
- Access to remote machine
  ssh bupt@210.25.132.137 (password: bupt)

10. Editor - vi

- Text editor
- Insert mode
- Override mode
- Use sub-commands
- Tradition tools and others

The `vi` command

- **`vi`** is a text editor.
- **`vi`** shows you part of your file and allows you to enter commands that change something (add new stuff, delete a char or line, etc).

`vi` modes

`vi` has a couple of *modes*:

- command mode: move the cursor around, move to a different part of the file, issue editing commands, switch to insert mode.
  - The command will not be displayed in screen
  - ENTER is not needed
- insert mode: whatever you type is put in the file (not interpreted as commands).

when you first start **`vi`** you will be in command mode.

73

Cursor Movement Commands

(only in command mode!)

`h`	move left one position
`l`	move right one position or space bar
`j`	move down one line
`k`	move up one line

Your arrow keys might work (depends on the version of vi and your terminal)

74

More Cursor Movement Commands

`$`	move to the end of the line
G	move to the end of the file
`w`	move forward one word
`b`	move backward one word
`e`	move to the end of the word
`)`	move to beginning of next sentence
`(`	move to beginning of current sentence

75

Scrolling Commands

`CTRL-F`	scroll forward one screen
`CTRL-B`	scroll backward one screen
`CTRL-D`	scroll forward 1/2 screen
`CTRL-U`	scroll backward 1/2 screen

76

Command that delete stuff

x	delete character (the one at the cursor)
X	delete back one character (backspace)
dw	delete word
dd	delete line
3x	delete 3 characters (any number works)
5dd	delete 5 lines (any number works)

Changing Text

cw	change word (end with Esc)
cc	change line (end with Esc)
C	change rest of the line
rx	replace character with 'x' (could be anything, not just 'x')

Insert Mode

- In insert mode whatever you type goes in to the file. There are many ways to get in to insert mode:

i	insert before current position
a	append (insert starting after cursor)
A	append at end of line
R	begin overwriting text
o	insert text in a new line below the current line
O	insert text in a new line above the current line

Copy & Paste

- P paste text last copied to the right of the cursor
- p paste text last copied to the left of the cursor
- yy copy current line
- ye copy from the cursor to the end of the word

Ending Insert Mode

- To get out of insert mode (back to command mode) you press "Esc" (the escape key).
- There is a status line (bottom of screen) that tells you what mode/command you are in.

Saving and Exiting

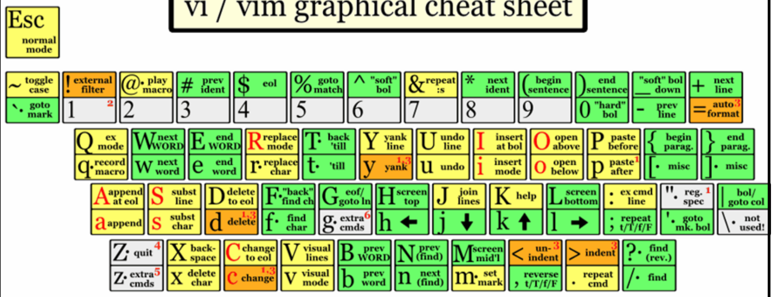
`ZZ`	save if changes were made, and quit.
`:wq`	Write file and quit
`:w`	Write file
`:w` ***`file`***	Write to file named ***`file`***
`:q`	Quit
`:q!`	Really quit (discard edits)

Searching and Replacing

`/text`	search forward for text
`?text`	search backward for text
`n`	repeat previous search
`N`	repeat search in opposite direction
`:s/findtext/`	replace the first occurrence of findtext
`:%s/findtext/`	replace all occurrences of findtext

version 1.1
April 1st, 06

vi / vim graphical cheat sheet



motion	moves the cursor, or defines the range for an operator
command	direct action command, if red, it enters insert mode
operator	requires a motion afterwards, operates between cursor & destination
extra	special functions, requires extra input
q·	commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line,
mk = mark, yank = copy

words: `quux(foo, bar, baz);`
WORDs: `quux(foo, bar, baz);`

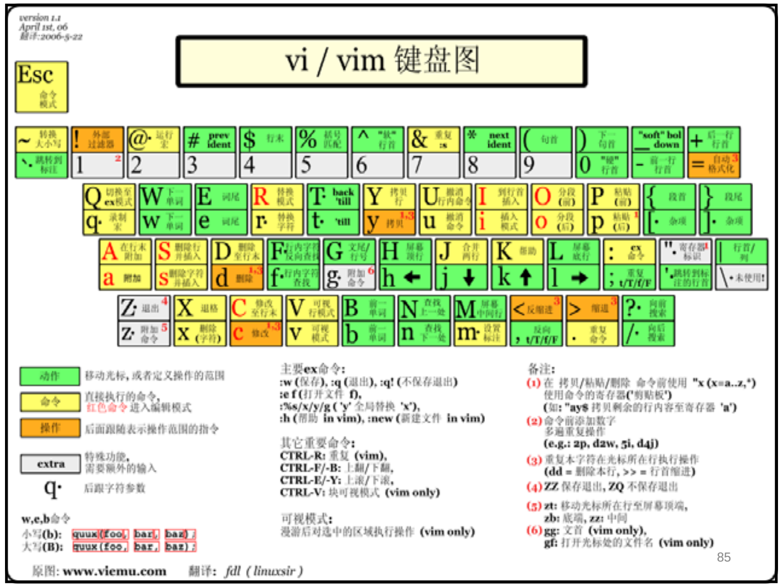
Main command line commands ('ex'):
:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:%s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :new (new file in vim),

Other important commands:
CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)

Visual mode:
Move around and type operator to act on selected region (vim only)

Notes:
(1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay$ to copy rest of line to reg 'a')
(2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)
(3) duplicate operator to act on current line (dd = delete line, >> = indent line)
(4) ZZ to save & quit, ZQ to quit w/o saving
(5) zt: scroll cursor to top, zb: bottom, zz: center
(6) gg: top of file (vim only), gf: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to **www.viemu.com** - home of ViEmu, vi/vim emulation for Microsoft Visual Studio



动作 移动光标，或者定义操作的范围
命令 直接执行的命令，红色命令 进入编辑模式
操作 后面跟随表示操作范围的指令
extra 特殊功能，需要额外的输入
q· 后跟字符参数

w,e,b命令
小写(b): quux(foo, bar, baz);
大写(B): quux(foo, bar, baz);

主要ex命令：
:w (保存), :q (退出), :q! (不保存退出)
:e f (打开文件 f),
:%s/x/y/g ('y' 全局替换 'x'),
:h (帮助 in vim), :new (新建文件 in vim)

其它重要命令：
CTRL-R: 重复 (vim),
CTRL-F/-B: 上翻/下翻,
CTRL-E/-Y: 上滚/下滚,
CTRL-V: 块可视模式 (vim only)

可视模式：
漫游后对选中的区域执行操作 (vim only)

备注：
(1) 在 拷贝/粘贴/删除 命令前使用 "x (x=a..z,*) 使用命令的寄存器('剪贴板') (如: "ay$ 拷贝剩余的行内容至寄存器 'a')
(2) 命令前添加数字 多遍重复操作 (e.g.: 2p, d2w, 5i, d4j)
(3) 重复本字符在光标所在行执行操作 (dd = 删除本行, >> = 行首缩进)
(4) ZZ 保存退出, ZQ 不保存退出
(5) zt: 移动光标所在行至屏幕顶端, zb: 底端, zz: 中间
(6) gg: 文首 (vim only), gf: 打开光标处的文件名 (vim only)

原图: www.viemu.com 翻译: *fdl* (*linuxsir*)

11. Other commands

- netstat
- arp
- ping
- traceroute
- ftp
- ps
- top
- df
- w
- last
- history
- … …

Change File Access Permissions

- **chmod** change the access permission of reading, writing and executing for file or directory
- chmod [who][op][mode] filename
  chmod [who][op][mode] directory
- Two manners: symbolic and numerical

Change File Access Permissions

- Symbolic manner
  who:
    u-user, g–group, O-other, A-all
  op:
    + - Adding the access permission defined by [mode]
    - - Deleting the access permission defined by [mode]
    = - Assigning the access permission defined by [mode]
  mode:
    r-read, w–write, x-execute
- Examples

```
$ chmod a+rx test.txt
$ chmod go-rx filename
```

-rw-rw-r--

Chmod a+x test.txt

Chmod a=x test.txt

Change File Access Permissions

- Numerical manner
  - Three octal numbers are used to describe the access mode
  - Three numbers for user, group, other
  - Examples:

```
$ chmod 741 test.txt
```

Change File Owner and Group

- **`chown`** change the owner of the file or directory

```
chown username filename
chown -R username directory
```

- Only the owner and the root user can change the owner of the file

Search for a File

- **`find`** find the files in the given directory according to the given expression

```
find pathname [option] expression
```

- option

```
-name       file name
-user       user name
-group      group name
-mtime n    files modified in n days
-newer fn   files modified later than fn
```

- Examples

```
$ find . -name test
$ find . –name ‘*abc*’
```

Locate a Command

- **Whereis** locate the binary, source and manual of the given command
  `whereis command`
- Examples
  `$ whereis ls`

Compiler for C in Linux

- **gcc**
- pre-process and compile
  `gcc –c hello.c`
  `hello.o` is created
- **link**
  `gcc –o hello hello.o`
  `hello` is created
- Execute (the absolute or relative path is needed)
  `./hello`
- **Compile and link can be done in one step**
  `gcc –o he hello.c`

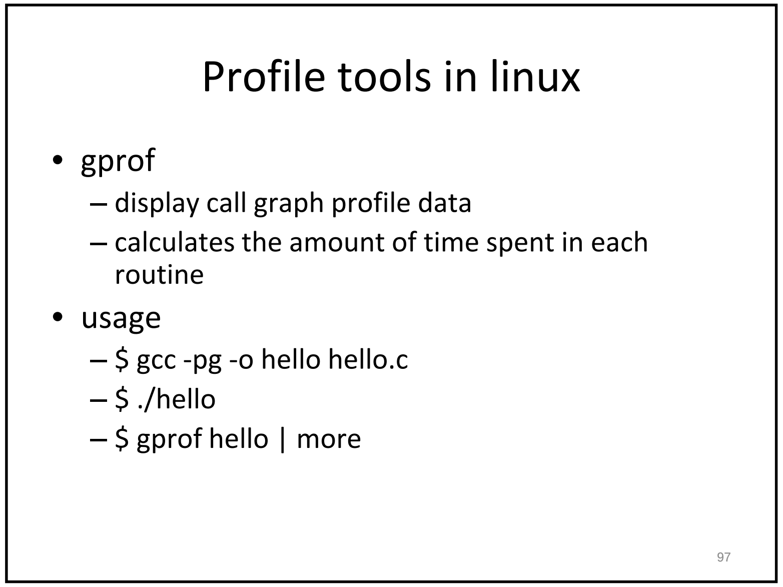
Interactive debugging in Linux

- **GDB（The GNU Project Debugger）**
  - r run;
  - b breakpoint;
  - p print;
  - n next;
  - s step into;
- **Usage**

```
gcc –g –o hi hello.c
gdb ./hi
```

Off-line debugging in Linux

- **Coredump**
  - **Recorded state of a program’s working memory when it crashed.**
  - **Used to capture memory status during dynamic memory allocation**
  - **Good at tracking a bug difficult to reappear**
- **Usage**
  - `gcc –g –o hi hello.c`
  - ulimit -c unlimited
  - ./hi （segment fault）
  - **`gdb ./hi core.222`**



Profile tools in linux

- gprof
  - display call graph profile data
  - calculates the amount of time spent in each routine
- usage
  - $ gcc -pg -o hello hello.c
  - $ ./hello
  - $ gprof hello | more

97