Languages for Informatics

1 – Introduction to UNIX and Shell

Department of Computer Science
University of Pisa
Largo B. Pontecorvo 3
56127 Pisa

Topics

- Linux programming environment (2h)
- Introduction to C programming (12h)
- Basic system programming in Linux (10h)

Overview

1. Introduction to Unix and Shell
   - Brief History
   - LINUX Architecture
   - Shell
   - the UNIX file system
   - Hard and Softlinks
   - Some useful commands
2. Installation guide for Arch Linux XUbuntu

Brief History (1)

MULTICS - In the late 1960, General Electric, MIT and Bell Labs launched a joint project, to develop a multi-user, multi-tasking OS called Multiplexed Information and Computing System MULTICS.

UNICS - inspired Ken Thompson at Bell Labs, to develop a simpler OS, Uniplexed Information and Computing System (UNICS) in 1969.

Brief History (2)

UNICS (shortend to UNIX) was designed with the following features in mind:

- programming environment;
- easy UI;
- simple utilities to be combined to create more powerful ones;
- hierarchical file system (tree-like);
- simple interfaces with devices;
- *multi-user* and *multi-process*: many users can connect simultaneously to the system and run processes;
- architecture-independent and transparent to the user.
- short space efficient commands, e.g. `ls`, `cp`, `mv`, etc.

Brief History (3)

1973 - Unix rewritten (mostly) in **C**, a high level language developed by **Dennis Ritchie**.

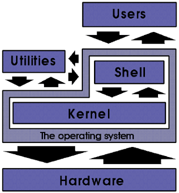
1974 - Unix was released to universities (academic license).

1978 - Split into two main branches, Bell Labs' System V and Berkeley Software Distribution (BSD).

1991 - Finnish undergraduate student Linus Torvalds creates a *unix-like* kernel (following the Single Unix Specification) for PCc and he calls it **Linux**. Linux kernel is included in GNU

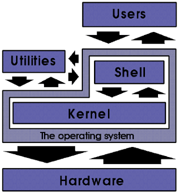
- SYS V-style startup files, BSD style system layout
- complies with a family of IEEE standards called POSIX (Portable Operating System Interface).
- open source - anyone can add features and correct deficiencies.
- several development streams: Debian, Ubuntu; Redhat, Fedora, CentOs; Slackware, SUSE.

LINUX OS Architecture (1)



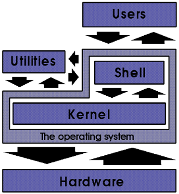
- Unix is organised in *layers*;
- The *Kernel* allows users' programs to access physical resources (memory, CPU, I/O);
- The file system is a hierarchical organisation of files and directories; the topmost level is the **root**;
- Users' programs interact with the kernel through *system calls*.

LINUX OS Architecture (2)



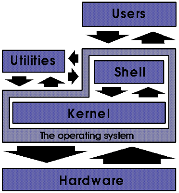
- The OS **kernel** controls underlying hardware. The kernel provides low-level device, memory and processor management functions such as HW interrupts, allocating memory for programs, sharing processor among programs,...

LINUX OS Architecture (3)



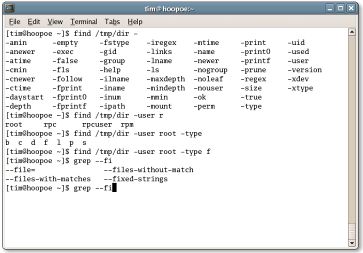
- HW independent Kernel services are exposed to higher layer program through a standard **library of system calls** described in the POSIX.1 specification (e.g. to execute a program, to create a file, open a logical network connection to another PC, etc.)

LINUX OS Architecture (4)



- Users’ **application programs** (e.g. calculator, mail client, media player, etc.) interact with the kernel through system calls.
- **utility programs** that come shipped with the OS (e.g. disk space analyzer, network analyzer, CPU info, etc.)
- **Daemons** provide remote network and administration services (e.g. `sshd`, `httpd`, `crond`, etc.)
- **Shells ...**

The UNIX Shell



```
[tim@hoopoe ~]$ find /tmp/dir -
-amin      -empty     -fstype    -iregex    -mtime     -print     -uid
-anewer    -exec      -gid       -links     -name      -print0    -used
-atime     -false     -group     -lname     -newer     -printf    -user
-cmin      -fls       -help      -ls        -nogroup   -prune     -version
-cnewer    -follow    -ilname    -maxdepth  -noleaf    -regex     -xdev
-ctime     -fprint    -iname     -mindepth  -nouser    -size      -xtype
-daystart  -fprint0   -inum      -mmin      -ok        -true
-depth     -fprintf   -ipath     -mount     -perm      -type
[tim@hoopoe ~]$ find /tmp/dir -user r
root     rpc      rpcuser  rpm
[tim@hoopoe ~]$ find /tmp/dir -user root -type
b  c  d  f  l  p  s
[tim@hoopoe ~]$ find /tmp/dir -user root -type f
[tim@hoopoe ~]$ grep --fi
--file=                --files-without-match
--files-with-matches   --fixed-strings
[tim@hoopoe ~]$ grep --fi
```

- A program that provides an interface to the functionalites of the OS
- interface is text-only or GUI
- It first reads commands from the user, and then it executes them (e.g., browse the file system, create files and directories, run programs).

Login

When Linux machine is accessed remotely through TeleTYpewriter terminal (TTY), you end up at the prompt

Shell

```
login as: MYNAME
login as: myname
login as: MyName
password:
$
```

Note

UNIX is case sensitive !!!

Login, Logout

To change your password, type in the TTY-shell

Shell

```
$ passwd
Changing password for MyName.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
```

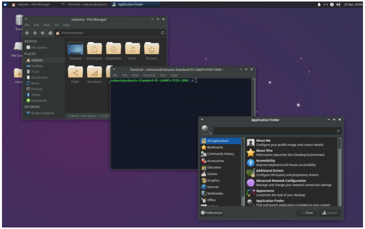
To logout, type

Shell

```
$ exit
```

Login, Logout (2)

When UNIX computer is accessed locally, a **graphical window manager** organizes placement and appearance of windows within a windowing system similar to MSWindows but with virtual desktops.



```
xubuntu@xubuntu-Standard-PC-i440FX-PIIX-1996:~$
```

About Me
Configure your profile image and contact details

About Xfce
Information about the Xfce Desktop Environment

Accessibility
Improve keyboard and mouse accessibility

Additional Drivers
Configure third-party and proprietary drivers

Advanced Network Configuration
Manage and change your network connection settings

Appearance
Customize the look of your desktop

Application Finder
Find and launch applications installed on your system

The launch a shell, look for icons mentioning `terminal`, `xterm`, `terminal emulator` or similar.

The file system (1)

UNIX OS is built around the concept of a **filesystem**, storing all long-term states of the system for logical organisation including

- OS kernel
- executable files supported by the OS
- configuration information
- temporary workfiles and user data
- special files for access to system hardware

as well as physical organisation, such as disk, cd-rom, dvd, etc.

Homogeneous : everything is a file (docs, sources, apps, images...). Four categories of files: *Ordinary files*, *Directories*, *Devices* and *Links*.

The file system (2)

Files & directories

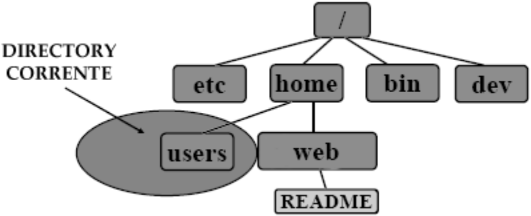
Each node of the tree is either a file or a dir, and a dir that contains other files and dirs.

- A **file** has non structured sequence of bytes (logical storage unit));
- A **directory** is a file that indexes other files.

A file, identified by a **path name**, has several attributes: type, access rights, owner, group, size, creation date, last change, last access.

The path name can be either **absolute**, starting from the root of the FS (/), or **relative**, relative to the current position of the user in the FS.

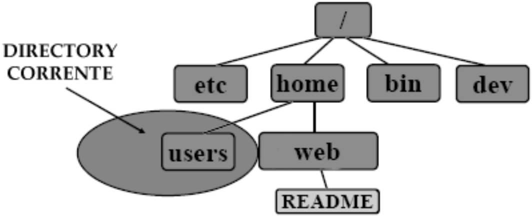
The file system (3)



In Unix, the file system has a *tree*-like structure (actually a *graph*)

- the topmost level is the root `'/'`.
- the current directory is `'.'`.
- the parent directory is `'..'`.
- the home directory is `'~'`.

The file system (3)



In Unix, the file system has a *tree*-like structure (actually a *graph*)

- **Absolute Name:** `/home/web/README`
- **Relative Name:** (w.r.t. users) `../web/README`
- (w.r.t. home) `./web/README` or `~/web/README`

Ordinary Files

- can contain text, data, program information
- cannot contain other files or directories
- UNIX filenames can contain any keyboard character except '/' up to 256 characters, including the specials '*','?','#','&' and whitespace character (but are **hard to use**).
- Best choice: alphanumerical characters, letters and numbers, combined by _ (underscore) and . (period).

Wildcards

Multiple filenames can be specified using special pattern-matching characters

- ’?’ matches **any single** char in that position in the filename
- ’*’ matches **all** chars in the filename
- `’*<exp>*’` matches **all** initial and final chars in the filename with `exp` in between
- `’[<char>]’` matches the **range** of chars in the brackets.

For example, `ls ?a?` lists all 3-letter filenames with ’a’ in the middle

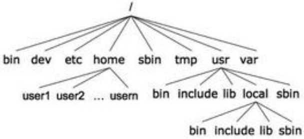
`ls *abc*` ... with ’abc’ in the middle

`ls [a-e]*` ... starting with a,b,c,d,e

`ls *.[xyz]*` ... that have an extension beginning with x, y or z.

Directories (1)

Main directories

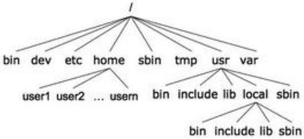


System directories, in all unix-like systems:

- **/:** Root directory
- **bin:** Essential low-level system executables (by user);
- **lib:** Program libraries (included by compiler) for low-level system utilities
- **usr:** Higher-level system utilities and apps
- **usr/bin:** - executables
- **usr/lib:** - libraries
- **sbin:** System executables (for system admin tasks by SU);

Directories (2)

Main directories



System directories, in all unix-like systems:

- **dev:** Hardware (*devices*);
- **etc:** UNIX system configuration and information files;
- **home:** dir containing the home dir of each user;
- **tmp:** Temporary files storage space;
- **var:** Logging and spooling.

Syntax of Unix commands

The typical syntax of a UNIX command is as follows:

```
command <options> <arguments>
```

- each command can ask the kernel the execution of a specific action;
- commands are binary files, executables by users.

`<options>` are not mandatory and act on the behaviour of the command. Usually they are specified using “-” in front of a letter.

`<arguments>` there might be arguments or not, depending by the command.

Documentation on commands

- `man command`: show the man page of `command`, with detailed instructions on its use and available options; e.g., `man ls`;
- `man -k word`: look for man pages that contain “word”; e.g., `man -k cat`;
- `apropos word`: look for string ’word’ in the description section of the man pages of all Unix commands. Useful to search for the exact name of a command that executes the action ’word’;
- `whatis command`: describes the function of `command` (apropos searches the *whatis* database for strings);
- `command --help`.

Change Directory

`cd [<dir>]` can be used to move across directories.

The parameter **`<dir>`** is optional — if not used, the command moves you to the home directory.

Example

- Assume we want access our personal docs in **`/home/user/docs`**
- assume the current dir is our home: **`/home/user`**
- to move to the docs dir execute:
  `cd docs`

List Directory (1)

```
ls [-alsFR] [<dir1> ...  <dirN>]
```

If no dir is given, it refers to the current directory.

Some of the available options:
-a show the hidden files (nome begins with “.”);
-l show extended info on files (e.g., rights, size, owner, group);
-s show size in bytes;
-F add a character at the end of the file name, to show its type (e.g., ”name/” denotes a directory);
-R show recursively the subdirectories (it executes ls recursively on subdirs).

List Directory (2)

Example

```
$ ls -al
drwxrwxr-x 3 MyName MyGroup 4096 Jun 28 20:24
Data
```

- 1st char: `d` (directory), `-` file, `l` symbolic link
- file permissions (3x3 for owner/group/others),
- number of (hard) links (3) equal # of sub-dir + parent dir + itself,
- owner name (NyName),
- owner group (MyGroup),
- file/directory size in bytes (4096),
- time of last modification (Jun 28 20:24).

Create a directory

```
mkdir [-p] <dir1> ...  <dirN>
```

The **dir** parameters denote the names (absolute o relative paths) of the directories to create.

Options:

-p create intermediate directories specified in the parameters **dir**.

Example

- **mkdir temp** — creates a directory **temp** in the current directory.
- **mkdir -p docs/personal** — creates the dir **personal** in dir **docs** (if **docs** does not exists it is created).

Delete a directory (when empty)

```
rmdir [-p] <dir1> ...  <dirN>
```

The `dir` parameters denote the names (absolute or relative path) of the directories to delete.

Options:

`-p` deletes intermediate directories specified in the parameters `dir`.

Example

- `rmdir temp` — delete directory `temp` if empty.
- `rmdir -p docs/personal` — delete directory `personal` and `docs`, if both empty.

Copy files

```
cp [-if] <file1> <file2>
```

copy **`file1`** in **`file2`** — if **`file2`** exists, it is overwritten!

```
cp [-if] <file1> ...  <fileN> <dir>
```

copy **`file`** in directory **`dir`** — if a **`file`** exists in **`dir`**, it is overwritten!

Options:
`-i` asks confirmation before overwriting;
`-f` no confirmation.

Move files

```
mv [-if] <file1> <file2>
```

move **`file1`** in **`file2`** — if **`file2`** exists, it is overwritten!

```
mv [-if] <file1> ...  <fileN> <dir>
```

move **`file`** in directory **`dir`** — if a **`file`** in **`dir`** exists, it is overwritten!

Options:
`-i` asks confirmation before overwriting;
`-f` no confirmation.

Show content of a file

```
cat [-nve] <file1> ...  <fileN>
```

Options:
-n each line is numbered;
-v show the non printable characters, beside newline, tab and form-feed (a special character to terminate a page on a printer);
-e show $ at the end of each line (when used with option **-v**);

`cat file1 file2 file3` concatenates the content of files (in the given order) showing their content;

Echo

The **`echo <arg>`** displays a line of text.

Example

```
$ echo Ciao!
Ciao!
```

The **`echo`** utility displays the contents of a variable.

Example

```
$ x=10
$ echo $x
10
```

Echo (2)

Note

In combination with wild-cards, echo lists files and directories !

Examples

```
$ ls
data-new data1 data2 inittab example1.txt

$ echo data*
data-new data1 data2

$ echo data?
data1 data2
```

Echo - Redirection

By default, Unix commands take the input from the keyboard (**standard input - stdin**) and send the output and error messages (if any) to the screen (**standard output - stdout**, **standard error - stderr**). Input/output in Unix can be redirected to/from (other) files, as follows:

Metacharacter	**Significance**
>	output redirection (new)
>>	output redirection (append)
<	input redirection (new)
<<	input redirection from command line

Echo - Redirection (2)

Example

```
$ echo programming in C > file.txt
$ cat file.txt
programming in C
$ echo makes fun >> file.txt
$ echo $(<file.txt)
programming in C makes fun
$ cat list1 list2 > biglist
$ sort biglist > sortbiglist
```

Echo - Quotes

We have seen the special characters ’?’,’*’ and ’$’. There is another type in UNIX: **Single backward quotes (`).** Commands within backward quotes are executed and their output substituted into that location.

Example

```
$ hostname
Desktop-INF
$ echo my machine is called `hostname`
my machine is called Desktop-INF
```

Pipe

The character “**|**” (pipe)

- takes the output of one command and feeds it into the following command (not file)
- The output of the last command is the output of the pipeline (by default it goes to the standard output).

The usage is **`command1 | command2`** .

Example

```
ls | more
```

Shows the output of **`ls`** (all files) one page at the time.

Devices

The `'/dev/'` location hosts the device files. The majority of devices are either

- **block** devices, storing or holding data
- **character** devices transmitting or transferring data

For example,

./sda1	First harddisk partition
./ttyS0	First serial port (mouse, modem)
./scd0	First SCSI CD-ROM device
./psaux	PS/2 connection (mice, keyboards)
./dsp	Audio devices
./js0	Standard gameport joystick

Links (1)

Hardlinks

Direct **Hardlinks** from one file to another can be created by

```
$ ln <source> <link>
```

- creates another directory entry for `source` called `link`.
- Both directory entries point to the same file.
- Both appear in the file permissions identical with link count of 2.
- If either `source` or `link` are modified, the change will be directly reflected in the other file.

Links (2)

Example

```
$ ln white whitelink
$ ls -l white*
-rwxrw-r-- 2 MyName MyGroup 17 set 25 09:57 white
-rwxrw-r-- 2 MyName MyGroup 17 set 25 09:57
whitelink
```

From the status of the link, we can see that

```
$ stat whitelink
File: 'whitelink'
Size: 17 Blocks: 8 IO Block: 4096 regular file
```

that the link is actually a regular file.

Links (3)

Consider the following scenario:

```
$ mkdir -p /tmp/a/b
$ cd /tmp/a/b
$ ln -d /tmp/a c
ln: failed to create hard link 'c' =>
'/tmp/a': Operation not permitted
```

A loop with back link c would have been created. Its depth were infinity:

```
$ cd /tmp/a/b/c/b/c/b/c/b/c/b
```

A file system with loops is no longer a tree. The unambiguity of parent tree directories is broken!

Links (4)

Softlinks

Softlinks can be created by

```
$ ln -s <source> <link>
```

The shortcut appears as an entry with a special type `’l’` in the file information. To see where the link points to, let us type

```
$ stat <link>
File:  ’<link>’ -> ’<source>’
Size:  5 Blocks:  0 IO Block:  4096 symbolic link
```

It becomes clear where the link points to.

Links (5)

Softlinks

```
$ ln -s white whitesoftlink
$ ls -l whitesoftlink
lrwxrwxrwx 1 MyName MyGroup 5 23 set 16.39
whitesoftlink -> white
$
```

- Link count of the source file remains unaffected.
- Permission bits on a symbolic link are unused (always `rwxrwxrwx`)
- Permissions on the `link` are determined by that on the `file`.
- When `source` is removed, `link` is invalid

Current Working Directory

Print current Working Directory (pwd) from the root

```
pwd [-LP]
```

Options:

-L –logical include symlinks [default]

-P –physical avoid all symlinks

Current Working Directory

Print current Working Directory (pwd) from the root

```
pwd [-LP]
```

Options:

-L –logical include symlinks [default]

-P –physical avoid all symlinks

Example

```
$ ln -s white link_to_white
$ cd link_to_white
$ pwd -L
/home/MyName/link_to_white
$ pwd -P
/home/MyName/white
```

File permissions (1)

File and directory permissions can only be modified by their owners, or by the superuser (`root`) according to

```
chmod -vR --preserve-root <permissions> <file>
```

Options:
-v output a diagnostic message for every file processed;
-R Change files **and** directories recursively.;
--preserve-root Do not operate recursively on '/'.

File permissions (2)

Permissions:

- may be specified as a sequence of 3 octal digits
- symbolically.
  - **u** (user), **g** (group), **o** (other), **a** (all)
  - **r** (read) [4], **w** (write) [2], **x** (execute) [1]
  - **+** (add), **–** (remove), **=** set.

For example,

```
$ chmod ug=rw,o-rw,a-x *.txt
```

sets the permissions on all files ending in *.txt to `rw-rw----`

And this ?

```
$ chmod 660 *.txt
```

Group ownership

Users assigned to a certain group, **share privilege, security and access** in a multiuser system (such as Linux).
The **group ownership** of files or directories can be changed by

```
chgrp -R <new_group> <file/directory>
```

The group membership can also be changed recursively with the `-R` option.

Example

```
$ chgrp mystudents system_and_security
```

All students member of the group `mystudents` have access to the directory `system_and_security` at the **same** time.

Inspecting File Content (1)

While **`cat`** lists the whole file (potentially filling the console screen buffer),

- **`head -n`**
  shows the first `n` lines of a text file
  e.g., `head -10 example.txt` shows the first 10 lines of `example.txt`;
- **`tail`**
  - **`-n`** shows the last `n` lines of a text file
    e.g., `tail -10 example.txt` — shows the last 10 lines of `example.txt`;
  - **`-f`** continuously monitors the last few lines of a **possibly changing** file e.g. `/var/log/auth.log`.

Inspecting File Content (2)

When only the **file type** is of interest, we can use the utility

```
file <file>
```

returning a high-level description of what typ of file it appears to be:

Example

```
$ file lecture1.pdf
lecture1.pdf:  PDF document, version 1.4
$ file background.jpg
background.jpeg:  JPEG image data, JFIF
standard 1.01, aspect ratio, density 1x1,
segment length 16, baseline, precision 8,
2048x1371, frames 3
```

Finding Files and Apps (1)

If you roughly know the name of a file, its location can be extracted by

```
find -RLP <directory> -name <file>
```

starting from the directory rooted at `directory`.

Options:
-R operate on files and directories recursively;
-L traverse every symbolic link to a directory;
-P Do **not** traverse every symbolic link to a directory (default).

Finding Files and Apps (1)

If you roughly know the name of a file, its location can be extracted by

```
find -RLP <directory> -name <file>
```

starting from the directory rooted at `directory`.

Options:
`-R` operate on files and directories recursively;
`-L` traverse every symbolic link to a directory;
`-P` Do **not** traverse every symbolic link to a directory (default).

Example

```
$ find /usr/bin -name Foxit*
/usr/bin/FoxitReader
/usr/bin/FoxitReader.sh
```

Finding Files and Apps (2)

A faster way of **locating all files** whose names match a particular search string is

```
locate <filename>
```

For example when you want to search for all filenames that have ’FoxitReader’, type

```
$ locate *FoxitReader*
```

Other useful commands

- **gzip/gunzip** — compression/decompression of files e.g., `gzip example.txt` — returns the compressed file `example.txt.gz`;
- **bzip2/bunzip2** — compression/decompression of files;
- **tar** — create/extracts from archives;
- **zip/unzip** and **rar/unrar** — creation and extraction of compressed archives.

Quiz

What are the options used to list the contents of a `.tar` file?

1. `cvf`
2. `tvf`
3. `xvf`
4. `lvf`

Appendix - Installation guide for XUbuntu

During class we use the Debian based OS **XUbuntu** using the desktop environment XFCE (X Freakin' Cool Environment ☺).

1. **XUbuntu only**
   - Go to https://xubuntu.org/download/
   - Download latest LTS desktop iso image
   - Boot from DVD
2. **Virtual Box** – compatible with Windows 7, 8, 8.1, 10 and MacOS
   - Get Oracle VM Virtualbox (latest version): https://www.virtualbox.org/
   - Get XUbuntu image for Virtualbox (*.vdi, latest version): https://www.osboxes.org/xubuntu/
   - Start Virtualbox, Create New Virtual Machine, and choose option ”Use existing file”, pointing to above vdi-file.

Installation of Linux (2)

3. **Virtual Box**
   - Go to settings and choose folder for shared drive (`auto-mounted`).
   - Go to settings and set network adapter to `NAT`.
   - Boot image.
   - The login password is `osboxes.org`.