

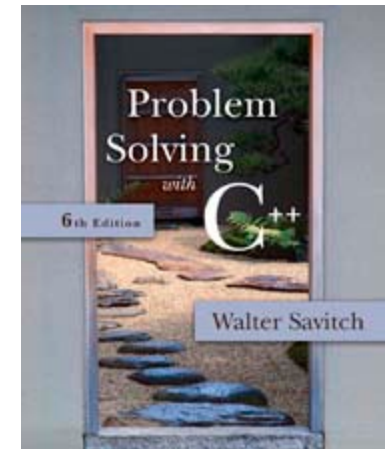
APS105: Lecture 33

Wael Aboelsaadat

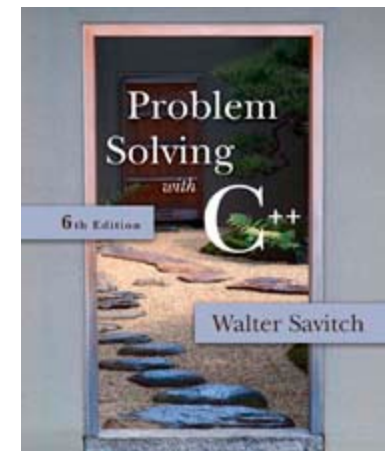
wael@cs.toronto.edu

<http://ccnet3.utoronto.ca/20079/aps105h1f/>

Acknowledgement: These slides are a modified version of the text book slides as supplied by Addison Wesley



UNIX



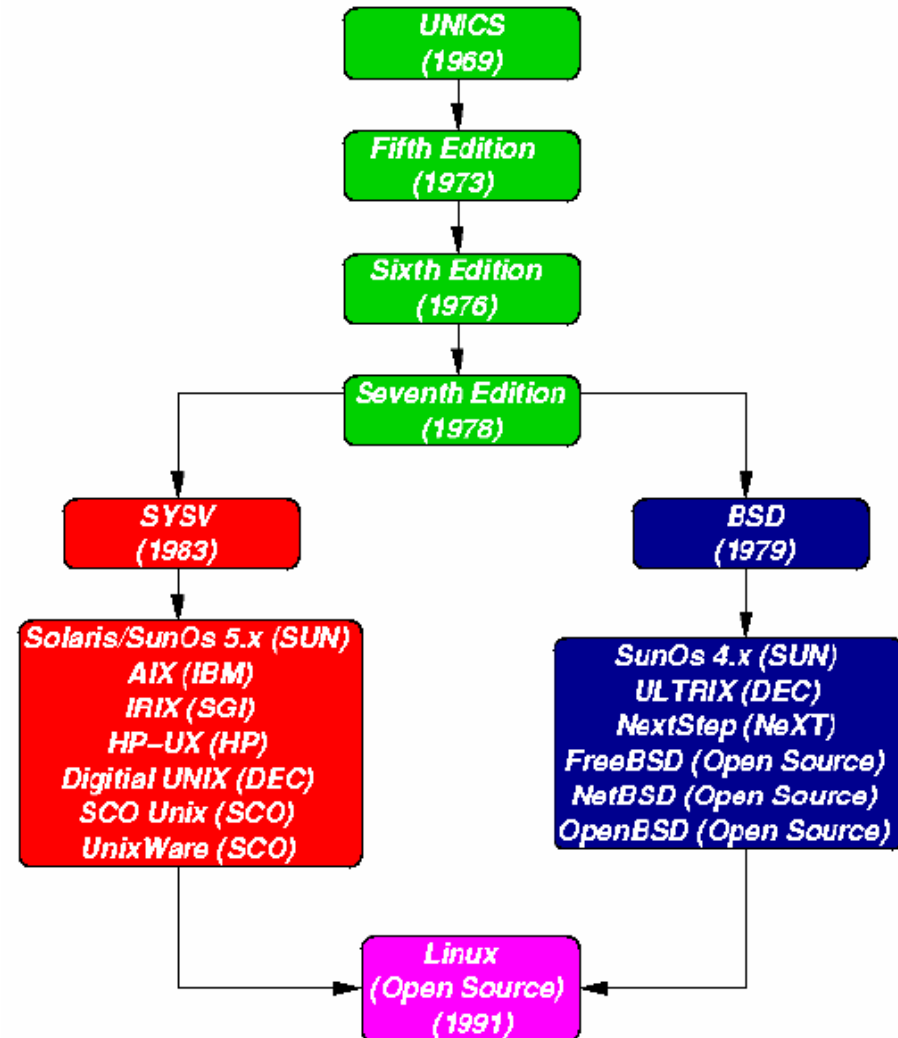
What is UNIX?

- UNIX is an operating system like Windows or Mac OS X
- From wikipedia:

An operating system (OS) is a set of computer programs that manage the hardware and software resources of a computer. An operating system processes raw system and user input and responds by allocating and managing tasks and internal system resources as a service to users and programs of the system. At the foundation of all system software, an operating system performs basic tasks such as controlling and allocating memory, prioritizing system requests, controlling input and output devices, facilitating networking and managing file systems.

UNIX History

UNIX is a “standard”,
i.e. there isn't ***a*** UNIX,
but multiple implementations
of the UNIX design
(Linux, BSD, Solaris,
AIX, OS X)

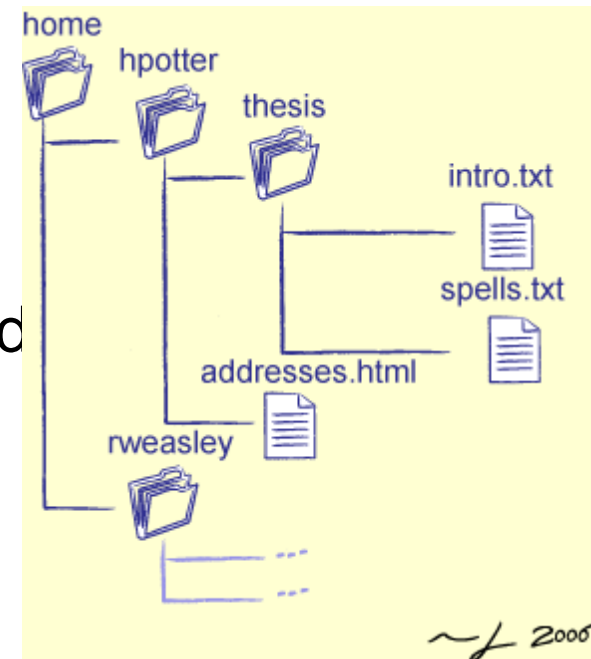


UNIX: files & programs

- file: collection of data
 - UNIX treats everything as a file (including peripherals)
- program:
 - collection of bytes representing code and data that are stored in a file
- process:
 - a program in execution (currently running, loaded from disk into RAM)

UNIX: file system

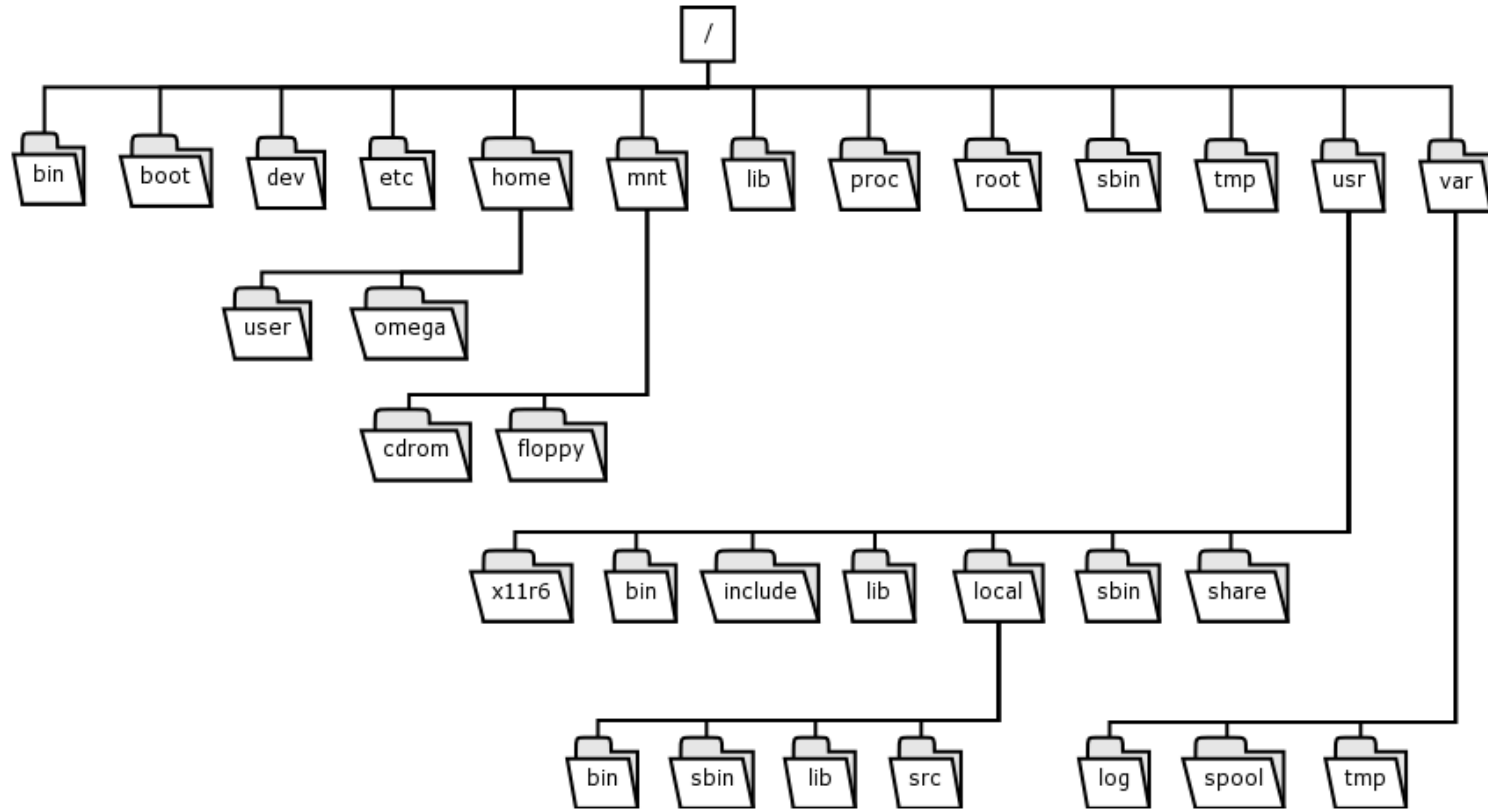
- The *file system* is the set of files and directories that the computer can access
- "Everything that doesn't go away when the computer is rebooted"
- Data is stored in files
- By convention, filename suffix identifies data type
 - E.g., .txt for text files, .mp3 for sound
 - But it *is* just a convention
 -



UNIX: file system

- On Unix, the file system has a unique root directory called / (pronounced "slash")
 - On Windows, every physical drive has its own root directory So `C:\home\hpotter\notes.txt` is different from `D:\home\hpotter\notes.txt`
- File and directory names are case-sensitive on most Unix variants, but case-*insensitive* on Windows
 - E.g., `This.txt` and `this.txt` are different files on Unix, but the same file on Windows
 - So you should never rely on casing to distinguish files

UNIX: file system directories



UNIX: file system navigation

- ***pwd*** (**present working directory**) shows the name and location of the directory where you are currently working:

```
> pwd
```

```
/u/browns02
```

- This is a “pathname,” the slashes indicate sub-directories
- The initial slash is the “root” of the whole filesystem

- ***ls*** (**list**) gives you a list of the files in the current directory:

```
> ls
```

```
assembin4.fasta Misc test2.txt
```

```
bin temp testfile
```

- Use the ***ls -l*** (**long**) option to get more information about each file

```
> ls -l
```

```
total 1768
```

```
drwxr-x--- 2 browns02 users 8192 Aug 28 18:26 Opioid
```

```
-rw-r----- 1 browns02 users 6205 May 30 2000 af124329.gb_in2
```

```
-rw-r----- 1 browns02 users 131944 May 31 2000 af151074.fasta
```

UNIX: file permissions

- Use the ***ls -l*** command to see the permissions for all files in a directory:

```
> ls -l
drwxr-x--- 2 browns02 users 8192 Aug 28 18:26 Opioid
-rw-r----- 1 browns02 users 6205 May 30 2000 af124329.gb_in2
-rw-r----- 1 browns02 users 131944 May 31 2000 af151074.fasta
```

- The username of the owner is shown in the third column. (The owner of the files listed above is **browns02**)
- The owner belongs to the group “**users**”
- The access rights for these files is shown in the first column. This column consists of 10 characters known as the attributes of the file: **r**, **w**, **x**, and **-**
 - r** indicates read permission
 - w** indicates write (and delete) permission
 - x** indicates execute (run) permission
 - indicates no permission for that operation

UNIX: file permissions

```
> ls -l
drwxr-x--- 2 browns02 users 8192 Aug 28 18:26 Opioid
-rw-r----- 1 browns02 users 6205 May 30 2000 af124329.gb_in2
-rw-r----- 1 browns02 users 131944 May 31 2000 af151074.fasta
```

- The first character in the attribute string indicates if a file is a directory (**d**) or a regular file (**-**).
- The next 3 characters (**rwX**) give the file permissions for the owner of the file.
- The middle 3 characters give the permissions for other members of the owner's group.
- The last 3 characters give the permissions for everyone else (others)
- The default protections assigned to new files on our system is: **-rw-r-----** (owner=read and write, group =read, others=nothing)

UNIX: Change Protections

- Only the owner of a file can change its protections
- To change the protections on a file use the ***chmod*** (**change mode**) command.

[Beware, this is a confusing command.]

- First you have to decide for whom you will change the access permissions:
 - the file owner (**u**)
 - the members of your group (**g**)
 - others (**o**) (ie. anyone with an RCR account)
- Next you have to decide if you are adding (+), removing (-), or setting (=) permissions.
- Taken all together, it looks like this:

```
> chmod u=rwx g+r o-x myfile.txt
```

This will set the owner to have read, write, and execute permission; add the permission for the group to read; and remove the permission for others to execute the file named **myfile.txt**.

UNIX: sub-directories

- ***cd*** (**change directory**) moves you to another directory

```
>cd Misc
> pwd
/u/browns02/Misc
```

- ***mkdir*** (**make directory**) creates a new sub-directory inside of the current directory

```
assembler phrap space
> mkdir subdir
> ls
assembler phrap space subdir
```

- ***rmdir*** (**remove directory**) deletes a sub-directory, but the sub-directory must be empty

```
> rmdir subdir
> ls
assembler phrap space
```

UNIX: file shortcuts

- There are some important shortcuts in Unix for specifying directories
 - `.` (dot) means "the current directory"
 - `..` means "the parent directory" - the directory one level above the current directory, so `cd ..` will move you up one level
 - `~` (tilde) means your Home directory, so `cd ~` will move you back to your Home.
 - Just typing a plain `cd` will also bring you back to your home directory

Unix: Processes and Multitasking

- To run a program in the background, use the “&” character (or “^Z” followed by “bg”):

```
> myprogram &  
[1] 7895
```

- **myprogram** is now running in the background as process id (PID) 7895
- Whenever your process finishes, it will print “Done” to the console.

Unix: Processes and Multitasking

- To check on the status of your jobs running on the system, use the **ps** command

```
> ps -a
      PID TTY          TIME CMD
      8095 pts/3        00:00:00 ps
```

- You can get an expanded list by typing **ps aux**, or by using the **top** command
- Use **uptime** to check the load average (how hard system is working) on slowly responding machines

Helpful Unix Commands

cat	cd	clear	cp
date	diff	echo	head
ls	man	mkdir	more
mv	od	passwd	pwd
rm	rmdir	sort	tail
uniq	wc	which	< >

Text Editors

- Become expert in either one of the following
- *emacs* (very popular)
 - Has menu bar like MS word along with keyboard shortcuts
 - <http://www.ucc.ie/doc/editing/emacs.html>
- *vi* (less commonly used) → *vim* is a more robust version
 - <http://www.linux.org/lessons/beginner/l5/lesson5c.html>

Features that made UNIX a hit

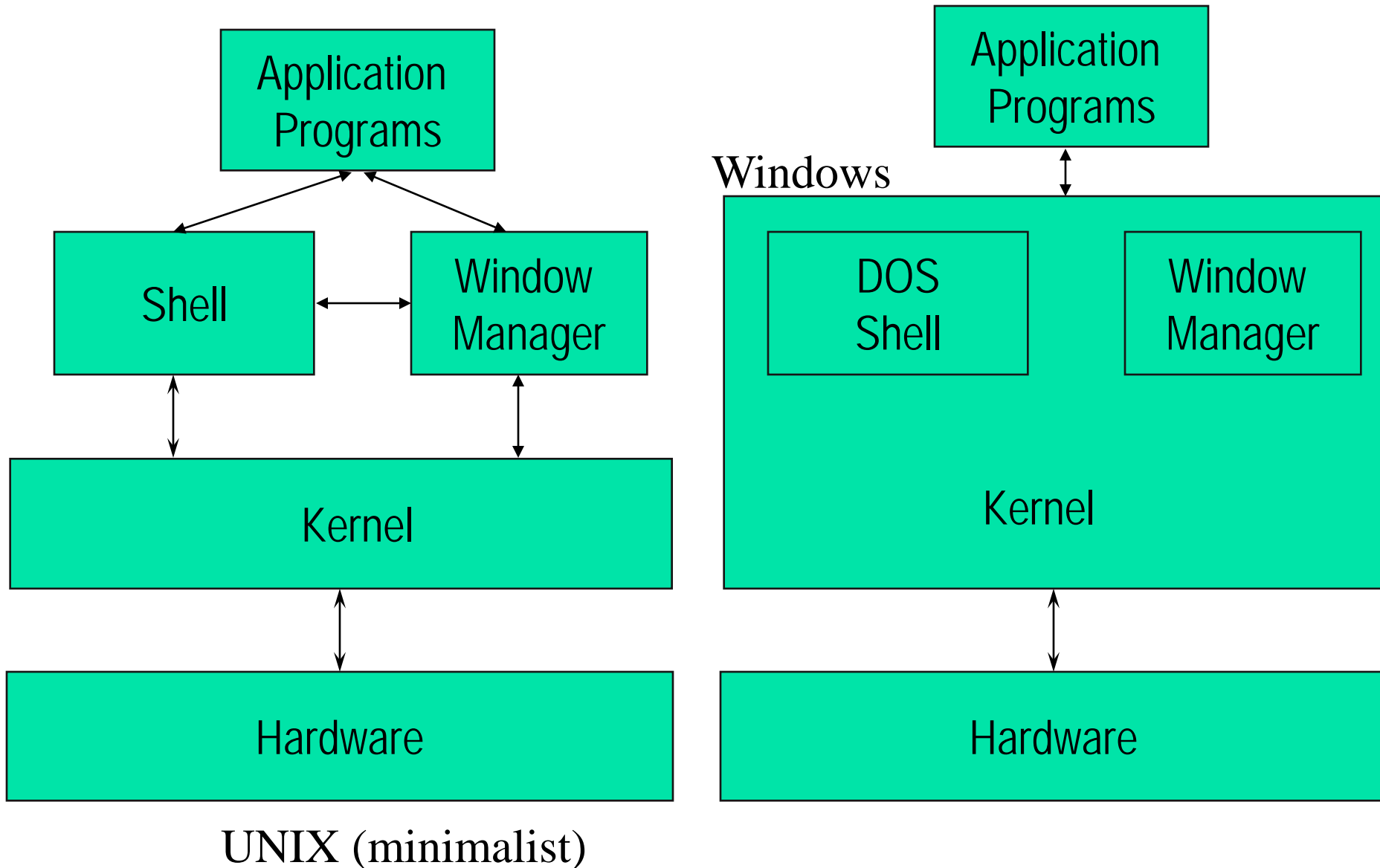
- Multitasking capability
- Multi-user capability
- Portability
- UNIX programs (tools, utilities)
- Library of application software

UNIX philosophies

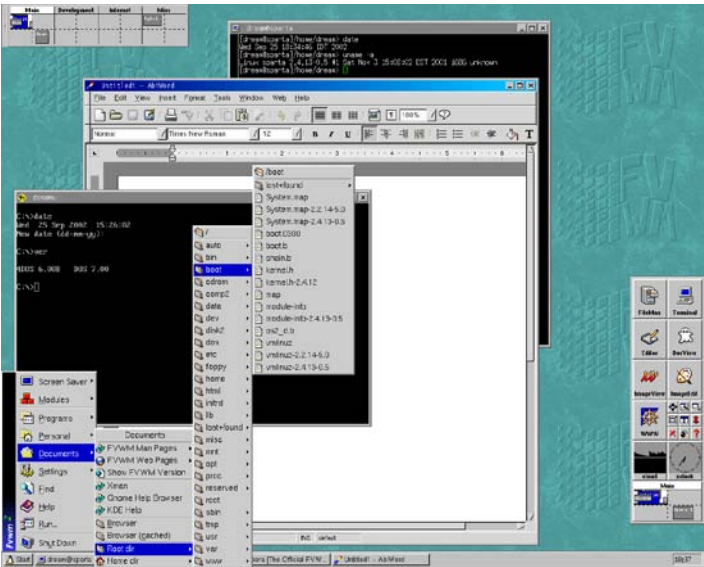
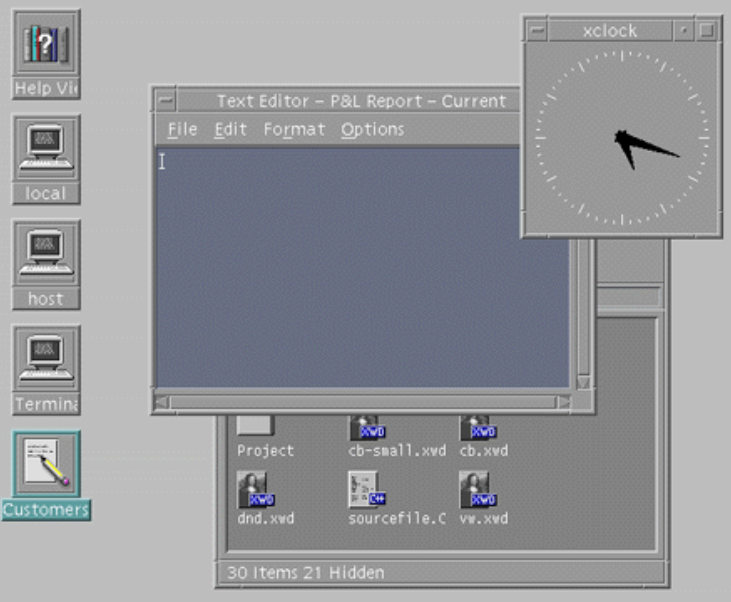
- How software should be written???
- a program should do one thing only
- it should do it well
- complex tasks should be performed by using programs together

combining small building blocks to make larger one...

UNIX vs. MS Windows architecture

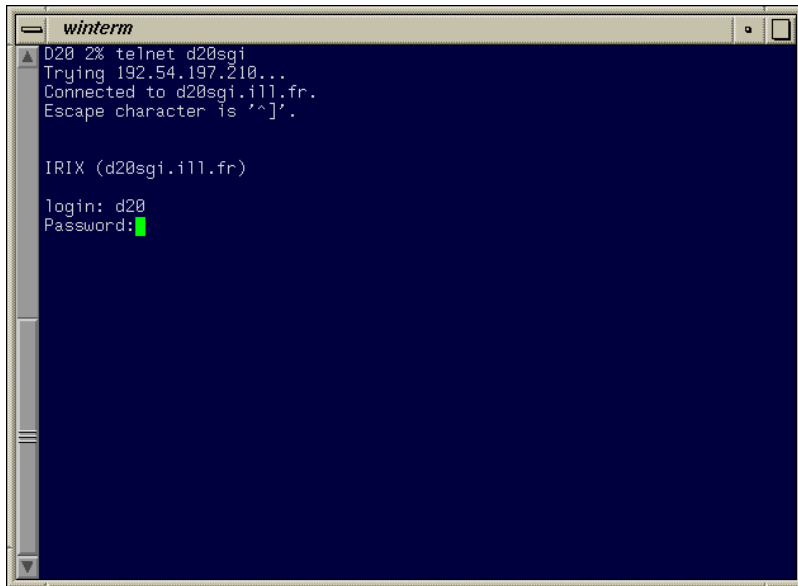


Why is UNIX better?



What is a Shell?

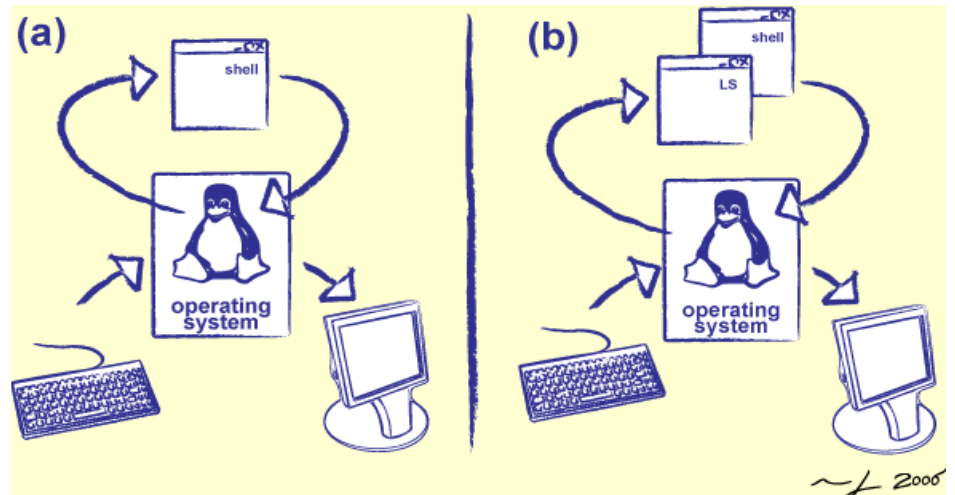
- ... a UNIX shell is a program that accepts and interprets commands and then has the operating system execute them.



```
winterm
D20 2% telnet d20sgi
Trying 192.54.197.210...
Connected to d20sgi.i11.fr.
Escape character is '^['.

IRIX (d20sgi.i11.fr)

login: d20
Password: █
```



- It is *not* the operating system

Shell: the Execution Cycle

- When you type a command, the operating system:
 - Reads characters from the keyboard
 - Passes them to the shell
- The shell:
 - Breaks the line of text into words
 - Looks for the program identified by the first word
 - Runs it, passing in the other words as arguments
 - Sends its output to the OS
- The OS displays the output in the current window

Flavors of Unix Shells

- Two main flavors of Unix Shells
 - Bourne (or Standard Shell): sh, ksh, bash, zsh
 - Fast
 - \$ for command prompt
 - C shell : csh, tcsh
 - better for user customization and scripting
 - %, > for command prompt

Shell Startup files

- **sh,ksh:**
 - profile (system defaults)
 - .profile
- **bash:**
 - .bash_profile
 - .bashrc
 - .bash_logout
- **csh:**
 - .login: executed when you logon
 - .cshrc: executed when a new shell is spawned
 - .logout: executed at logout

UNIX: online help

- man
 - Detailed description of a command
- info
 - More complete descriptions of certain packages
- help
 - Display helpful information about builtin commands
- apropos
 - Search the manual page names and descriptions
- whatis
 - display manual page descriptions

UNIX: Where To Go

- Unix Tutorial for Beginners



Deboray S. Ray and Eric J. Ray: *Unix*.
Peachpit Press, 2003, 0321170105.