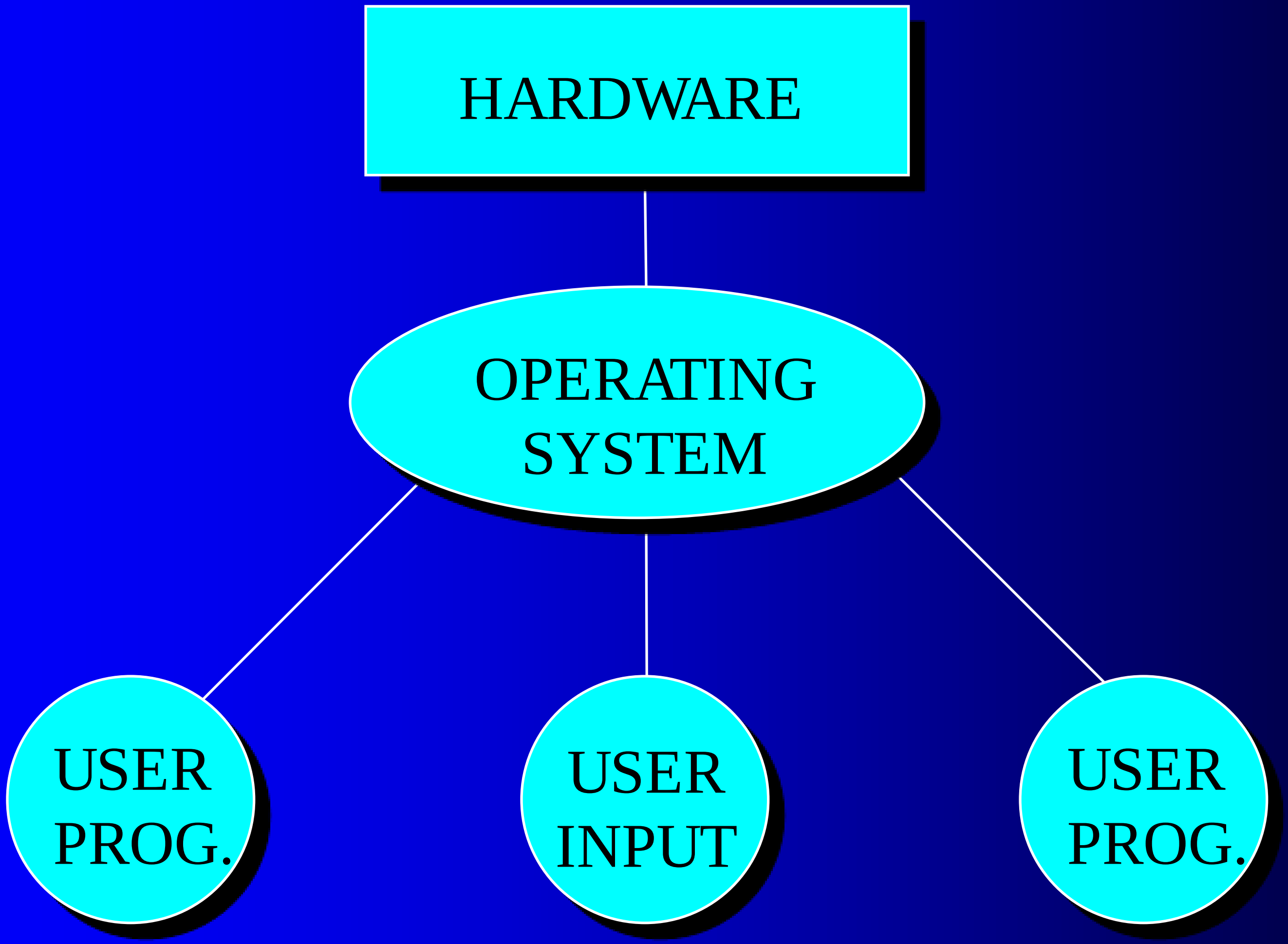
Lecture 2

- Covers
  - Operating systems
  - The Unix operating system
  - Compiling and running Java programs

- Reading: Hahn, Student Guide to Unix

- Operating systems

Operating systems



The operating system

- Is a resident program (runs all the time)
- Performs two important functions
  - Provides the interface between the user and the computer
  - Manages the computer’s resources: CPU time, memory space, file organisations

Thus …

- The OS functions as a critically important layer between the user and the machine
- It provides
  - Means to take requests from the user
  - Means to access files and programs
  - Ways to start and swap between programs
  - Ways to create new programs

    (together with editors/word processors and compilers)

Examples

- MS Windows
- MS DOS
- Unix
- VAX/VMS

- The Unix operating system

The Unix operating system

- Multitasking, multi-user OS
- The name UNIX
  - Is used in reference to a specific operating system branded to AT&T
  - Is also used in reference to a family of operating systems that meet a specific standard
  - This family includes Linux

Unix accounts

- A user is a person with an account on a machine
- A userid or username is a unique name for a user’s account on a machine
- Each account has a password which is a secret code required to access it
- An account has details associated with it such as an expiration date and an amount of disk space that it is allowed to use

Unix accounts

- Each account has a home directory where creating and deleting files and directories is allowed
- On initially logging into an account, the current working directory is set to the account's home directory
- To log out of an account use the command:

```
> logout
```

or

```
> exit
```

The Unix file system

- Within Unix, a file is any source of input or target of output
- There are 3 types of files
  - Ordinary (text or binary) files
  - Directories (contain other files)
  - Special (device) files
- The Unix file system is a tree-structured hierarchy, starting with the root directory /
- A file name can contain any character except /

Example of a Unix file system

Paths and filenames

- *Absolute pathname:* full name of every directory from the root to the actual file
- *Relative pathname:* starts from the current (working) directory
- Handy abbreviations in pathnames

..	parent directory
.	current or working directory
~	home directory

- Unix is case sensitive, i.e. it distinguishes between uppercase characters (A..Z) and lowercase characters (a..z)

Moving around the directory tree

```
> cd  <directory>
```

change directory

cd ~ will change to your home directory

```
> pwd
```

displays the current directory

Managing directories

> mkdir <directory>	make new directory
> rmdir <directory>	remove directory
> mv <directory> <target>	move directory
> ls	list contents of current directory
> ls <directory>	list contents of specified directory

Managing ordinary files

> cp <file1> <file2>	copy file
> cp <file1> <directory>	
> mv <file1> <file2>	move (or
> mv <file1> <directory>	rename) file
> rm <file>	remove file

Displaying files

```
> cat <file>
```

displays the file on the screen

```
> more <file>
```

displays one screenful at a time

(press the space bar to get the next screenful)

```
> less <file>
```

like the more command but more powerful

(can search with / and go backwards and forwards within the file)

Wild card characters

- The asterisk is a wildcard character
- It matches any sequence of characters, even an empty one
- Examples

```
> ls *.java
> ls Test*.java
> rm *.class
> rm *
```

Be very careful with this!

Wild card characters

- To specify characters from a set, enclose them in square brackets
- Examples

```
> ls [Aa]*.java
```

lists all files that start with uppercase A or lowercase a

```
> ls *[0-9]
```

lists all files that end with a numeral

Shells

- A shell is a program that accepts user input as commands and executes them
- There are various choices of shell in Unix
  - Bourne Shell (sh)
  - Korn Shell (ksh)
  - Bourne Again Shell (bash)
  - C-Shell (csh)
  - Tcsh (tcsh)
- Each shell has a slightly different look and feel

Tcsh

- We will use the tcsh
- When you start a shell, you can customise it to your liking
- Place customisation commands in the file called .cshrc in your home directory
- You can place other customisation commands in the .login and .logout files which are executed once when you log into or out of an account

Shortcuts to enter commands

```
> history
```

shows the last commands you have entered (saved in the history list)

```
> !!
```

repeats the last command you entered

```
> !<number>
```

executes command number <number> from the history list

Shortcuts to enter commands

```
> !<pat>
```

executes the last command starting with <pat>

```
> !?<pat>?
```

executes the last command containing <pat>

```
> ^<pat>^<new-pat>
```

repeats the last command substituting <new-pat> for <pat>

Shortcuts to enter commands

- <tab> completes a filename or command
- <ctrl-d> shows you the possibilities for a filename or command

Managing processes

- A process is a program that is running (or executing)
- <ctrl-c> terminates the currently running foreground process

Selected utilities

> date

displays the current time and date

> cal

displays a calendar for the current month

> cal <year>

displays a calendar for the given year

> cal <month> <year>

displays a calendar for the given month of the specified year

Selected utilities

> who

displays a list of the users currently logged in

> whoami

displays the userid of the account logged in

> finger

displays information about users logged in

> finger <userid>

Selected utilities

```
> finger <user's family or given name>
```

displays information about users with this name

```
> hostname
```

displays the name of the logged-into machine

Getting help

```
> man <command>
```

displays the online manual pages for the specified command

```
> apropos <topic>
> man -k <topic>
```

displays a list of the man pages about commands related to the specified topic

The vi editor

- The vi editor is a fully featured editing environment
- While it is not easy to learn initially, it pays dividends to put some time into learning it, as it will save you significant time and effort in developing programs

```
> vi filename
```

opens a file for editing in the vi editor

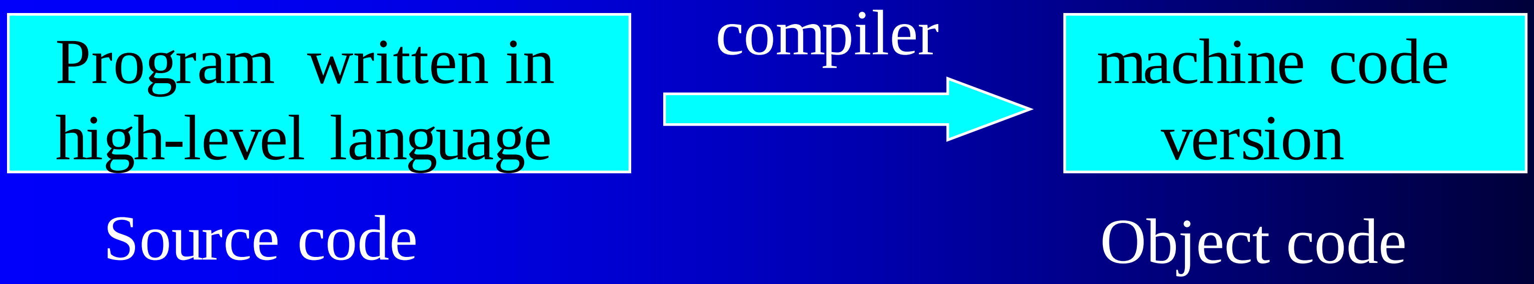
The vi editor

- vi has two modes: insert mode and command mode
  - Text is typed into a file in insert mode
  - Most other operations such as cutting and pasting occur in command mode
  - When vi starts it will be in command mode
  - To change from command mode into insert mode type ‘a’ or ‘i’ (or other similar commands)
  - To change from insert mode into command mode hit the <esc> key
  - To close the file and save changes type :wq in command mode
  - Refer to the list of vi commands supplied in the lab for other useful and powerful commands

► Compiling and running Java programs

Running high-level programs

- High-level language
  - Problem-oriented, must be translated to low-level
- Low-level language
  - What the machine actually executes
- Traditional compilation process

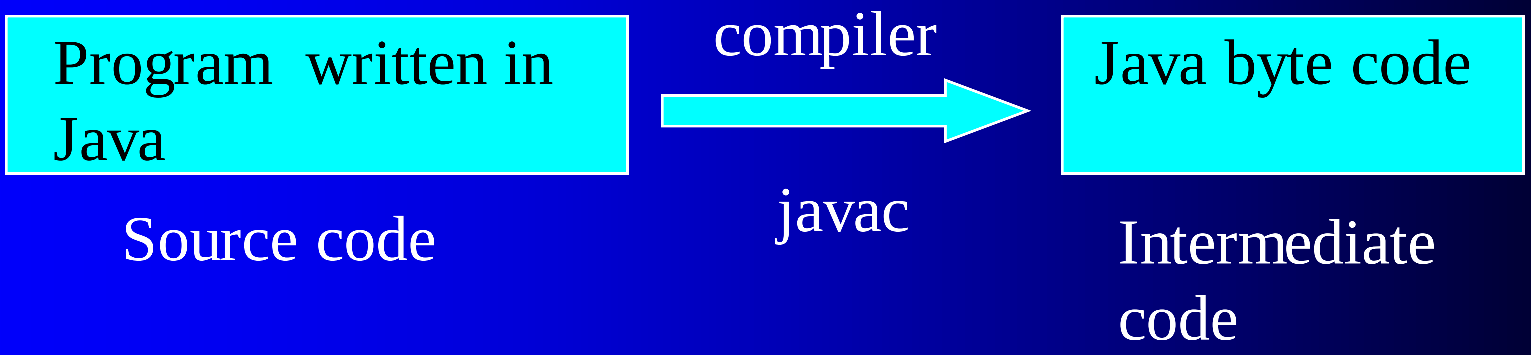


Byte code and JVM

- Programs written in high-level language are mostly translated into machine code, which is then directly executed by the CPU
- Java is an exception
- Java programs are translated into byte code, which is then executed by the Java Virtual Machine (JVM)
- The JVM is an interpreter program in machine code

Running Java programs

- Java compilation process



- Java execution process
  – Java byte code is read and executed via a Java byte code interpreter

Byte code and JVM

- The javac command converts the source programs into byte code
- Byte code files are those with the extension .class
- The java command causes the JVM to execute the byte code

(To increase execution speed, it is an option to convert the byte code into machine code)

Create, compile and run Java programs in Unix

- To create a file

```
> vi <filename>
```

Note: to create a Java program the filename must end in .java

- To compile a program

```
> javac <filename>.java
```

- To run the program

```
> java <classfilename>
```

Next lecture

- Object-oriented concepts