

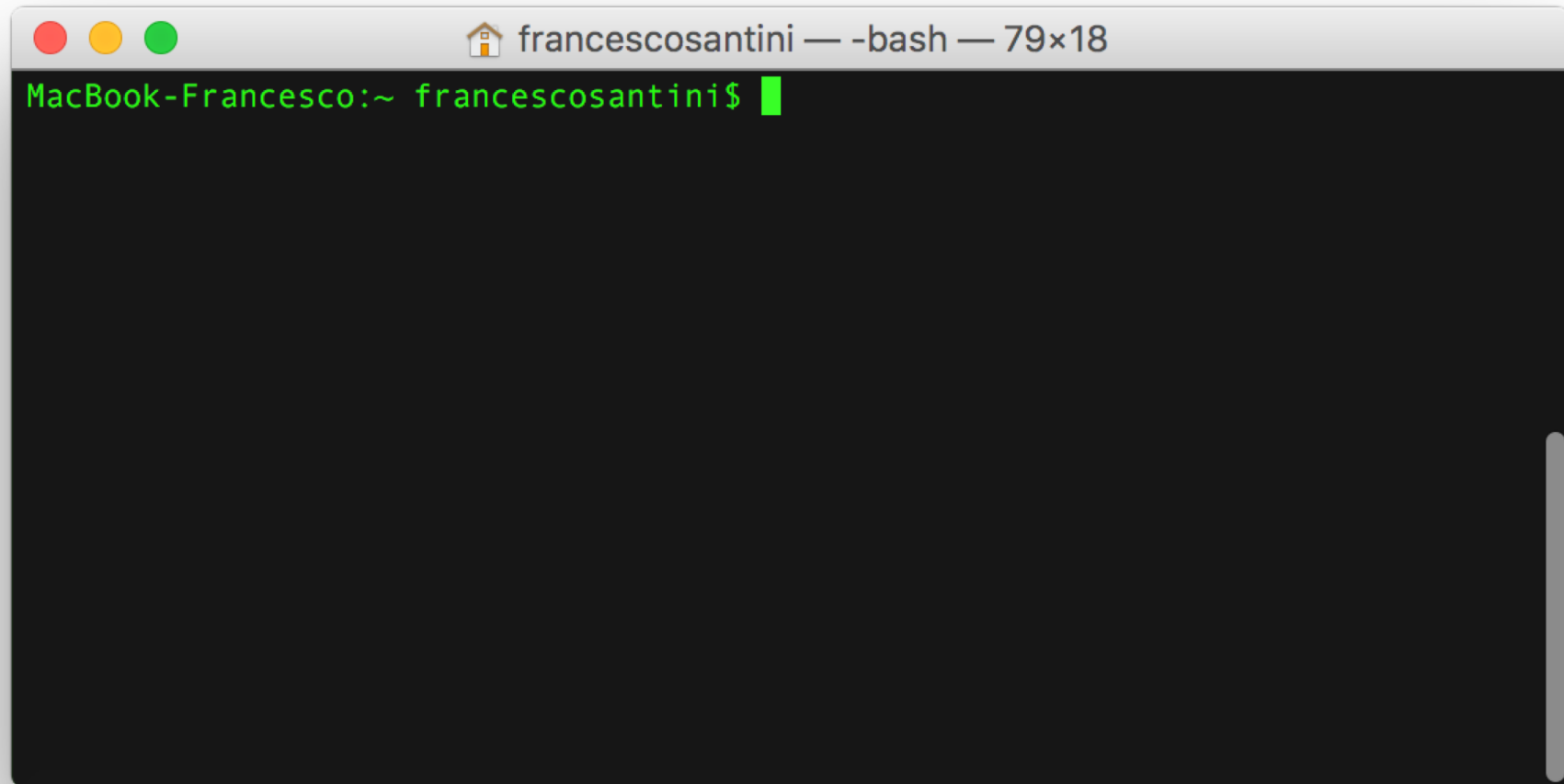
PROGRAMMAZIONE PROCEDURALE



SHELL



SHELL



SHELL

- ④ A program that interprets commands
- ④ Allows a user to execute commands by typing them manually at a terminal, or automatically in programs called *shell scripts*.
- ④ A shell is *not* an operating system. It is a way to interface with the operating system and run commands.

BASH

- ④ BASH = **B**ourne **A**gain **S**hell
- ④ Bash is a shell written as a free replacement to the standard Bourne Shell (/bin/sh) originally written by Steve Bourne for UNIX systems.
- ④ It has all of the features of the original Bourne Shell, plus additions that make it easier to program with and use from the command line.
- ④ Since it is Free Software, it has been adopted as the default shell on most Linux systems.

```
MacBook-Francesco:~ francescosantini$ echo $SHELL
/bin/bash
MacBook-Francesco:~ francescosantini$
```

BASH AND DOS COMMAND PROMPT

- ④ Case sensitivity: In Linux/UNIX, commands and filenames are case sensitive, meaning that typing “EXIT” instead of the proper “exit” is a mistake.
- ④ Filenames: The DOS world uses the “eight dot three” filename convention, meaning that all files followed a format that allowed up to 8 characters in the filename, followed by a period (“dot”), followed by an option extension, up to 3 characters long (e.g. FILENAME.TXT). In UNIX/Linux, there is no such thing as a file extension. Periods can be placed at any part of the filename, and “extensions” may be interpreted differently by all programs, or not at all.

SPECIAL CHARACTERS

<i>Character</i>	<i>Description</i>
<code>\</code>	Escape character. If you want to reference a special character, you must “escape” it with a backslash first. Example: <code>touch /tmp/filename*</code>
<code>/</code>	Directory separator, used to separate a string of directory names. Example: <code>/usr/src/linux</code>
<code>.</code>	Current directory. Can also “hide” files when it is the first character in a filename.
<code>..</code>	Parent directory
<code>~</code>	User's home directory
<code>*</code>	Represents 0 or more characters in a filename, or by itself, all files in a directory. Example: <code>pic*2002</code> can represent the files <code>pic2002</code> , <code>picJanuary2002</code> , <code>picFeb292002</code> , etc.
<code>?</code>	Represents a single character in a filename. Example: <code>hello?.txt</code> can represent <code>hello1.txt</code> , <code>helloz.txt</code> , but not <code>hello22.txt</code>
<code>[]</code>	Can be used to represent a range of values, e.g. <code>[0-9]</code> , <code>[A-Z]</code> , etc. Example: <code>hello[0-2].txt</code> represents the names <code>hello0.txt</code> , <code>hello1.txt</code> , and <code>hello2.txt</code>

SPECIAL CHARACTERS

	“Pipe”. Redirect the output of one command into another command. Example: <code>ls more</code>
>	Redirect output of a command into a new file. If the file already exists, over-write it. Example: <code>ls > myfiles.txt</code>
>>	Redirect the output of a command onto the end of an existing file. Example: <code>echo "Mary 555-1234" >> phonenumber.txt</code>
<	Redirect a file as input to a program. Example: <code>more < phonenumber.txt</code>
;	Command separator. Allows you to execute multiple commands on a single line. Example: <code>cd /var/log ; less messages</code>
&&	Command separator as above, but only runs the second command if the first one finished without errors. Example: <code>cd /var/logs && less messages</code>
&	Execute a command in the background, and immediately get your shell back. Example: <code>find / -name core > /tmp/corefiles.txt &</code>

EXECUTING COMMANDS

- ④ **The Command PATH:** Most common commands are located in your shell's "PATH", meaning that you can just type the name of the program to execute it.
 - ✓ Example: Typing "ls" will execute the "ls" command.
- ④ Your shell's "PATH" variable includes the most common program locations, such as /bin, /usr/bin, /usr/X11R6/bin, and others.
- ④ To execute commands that are not in your current PATH, you have to give the complete location of the command.
 - ✓ Examples: /home/esercizi/esercizio1
./esercizio1 (Execute a program in the current directory)

CHANGING PATH OF EXECUTABLES

```
MacBook-Francesco:~ francescosantini$ echo $PATH  
/opt/local/bin:/opt/local/sbin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/  
sbin:/opt/X11/bin:/usr/local/git/bin:/usr/texbin
```

```
MacBook-Francesco:~ francescosantini$  
PATH=$PATH:~/Desktop/Programml/  
MacBook-Francesco:~ francescosantini$ echo $PATH  
/opt/local/bin:/opt/local/sbin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin  
:/opt/X11/bin:/usr/local/git/bin:/usr/texbin:/Users/francescosantini/De  
sktop/Programml/
```

You can run a program with a.out instead of ./a.out

CHANGING PATH (PERMANENT)

- ② The command before does not work anymore if you close and restart the shell
- ② To make it permanent:
 - ✓ Modify the profile of your bash
 - Edit file: ~/.bash_profile
 - ✓ Add :~/Desktop/ProgrammI/ at the end of the string

COMMAND SYNTAX

📍 **Command Syntax:** Commands can be run by themselves, or you can pass in additional arguments to make them do different things. Typical command syntax can look something like this:

📍 `command [-argument] [--argument] [file]`

📍 Examples:

- ✓ `ls` List files in current directory
- ✓ `ls -l` Lists files in “long” format
- ✓ `cat filename` Show contents of a file
- ✓ `cat -n filename` Show contents of a file, with line numbers

```
MacBook-Francesco:~ francescosantini$ cat -n .bash_profile
1    PATH=$PATH:~/Desktop/Programml/
```

HELP!

- ④ Help on most Linux commands is typically **1)** built right into the commands themselves, **2)** available through online help programs (“man pages” and “info pages”).
- ④ ***Using a Command's Built-In Help:*** Many commands have simple “help” screens that can be invoked with special command flags. These flags usually look like “-h” or “--help”.
 - ✓ Example: `gcc --help`

HELP!

- ④ The best source of information for most commands can be found in the manual pages, known as “man pages” for short. To read a command's man page, type “man command”. Examples
 - ✓ man ls Get help on the “ls” command.
 - man man A manual about how to use the manual!
- ④ One you are in the manual, to search for a particular word within a man page, type “/word”.
- ④ To quit from a man page, just type the “Q” key.
- ④ Move up-down with arrows.
- ④ Type “h” for help INSIDE man.

MAN FOR STDLIBRARY

🕒 It works also for functions in standard library

MacBook-Francesco:~ francescosantini\$ man calloc

```
francescosantini — less ◀ man calloc — 91×27

MALLOC(3)                                BSD Library Functions Manual                                MALLOC(3)

NAME
  calloc, free, malloc, realloc, reallocf, valloc -- memory allocation

SYNOPSIS
  #include <stdlib.h>

  void *
  calloc(size_t count, size_t size);

  void
  free(void *ptr);

  void *
  malloc(size_t size);

  void *
  realloc(void *ptr, size_t size);

  void *
  reallocf(void *ptr, size_t size);

  void *
  valloc(size_t size);

:
```

NAVIGATING THE LINUX FILESYSTEM



ROOT DIRECTORY

- ④ The Linux filesystem is a tree-like hierarchy hierarchy of directories and files.
- ④ At the base of the filesystem is the “/” directory, otherwise known as the “root” (not to be confused with the root user).
- ④ Unlike DOS or Windows filesystems that have multiple “roots”, one for each disk drive, the Linux filesystem mounts all disks somewhere underneath the / filesystem.

MAIN DIRECTORIES

<i>Directory</i>	<i>Description</i>
	The nameless base of the filesystem. All other directories, files, drives, and devices are attached to this root. Commonly (but incorrectly) referred to as the “slash” or “/” directory. The “/” is just a directory separator, not a directory itself.
/bin	Essential command binaries (programs) are stored here (<code>bash</code> , <code>ls</code> , <code>mount</code> , <code>tar</code> , etc.)
/boot	Static files of the boot loader.
/dev	Device files. In Linux, hardware devices are accessed just like other files, and they are kept under this directory.
/etc	Host-specific system configuration files.
/home	Location of users' personal home directories (e.g. <code>/home/susan</code>).
/lib	Essential shared libraries and kernel modules.
/proc	Process information pseudo-filesystem. An interface to kernel data structures.
/root	The root (superuser) home directory.
/sbin	Essential system binaries (<code>fdisk</code> , <code>fsck</code> , <code>init</code> , etc).

MAIN DIRECTORIES

/tmp	Temporary files. All users have permission to place temporary files here.
/usr	The base directory for most shareable, read-only data (programs, libraries, documentation, and much more).
/usr/bin	Most user programs are kept here (<code>cc</code> , <code>find</code> , <code>du</code> , etc.).
/usr/include	Header files for compiling C programs.
/usr/lib	Libraries for most binary programs.
/usr/local	“Locally” installed files. This directory only really matters in environments where files are stored on the network. Locally-installed files go in <code>/usr/local/bin</code> , <code>/usr/local/lib</code> , etc.). Also often used for software packages installed from source, or software not officially shipped with the distribution.
/usr/sbin	Non-vital system binaries (<code>lpd</code> , <code>useradd</code> , etc.)
/usr/share	Architecture-independent data (icons, backgrounds, documentation, <code>terminfo</code> , man pages, etc.).
/usr/src	Program source code. E.g. The Linux Kernel, source RPMs, etc.
/usr/X11R6	The X Window System.
/var	Variable data: mail and printer spools, log files, lock files, etc.

COMMANDS FOR NAVIGATING

<i>Linux Command</i>	<i>DOS Command</i>	<i>Description</i>
pwd	cd	“Print Working Directory”. Shows the current location in the directory tree.
cd	cd, chdir	“Change Directory”. When typed all by itself, it returns you to your home directory.
cd directory	cd directory	Change into the specified directory name. Example: <code>cd /usr/src/linux</code>
cd ~		“~” is an alias for your home directory. It can be used as a shortcut to your “home”, or other directories relative to your home.
cd ..	cd..	Move up one directory. For example, if you are in <code>/home/vic</code> and you type “ <code>cd ..</code> ”, you will end up in <code>/home</code> .
cd -		Return to previous directory. An easy way to get back to your previous location!

COMMANDS FOR NAVIGATING

<code>ls</code>	<code>dir /w</code>	List all files in the current directory, in column format.
<code>ls directory</code>	<code>dir directory</code>	List the files in the specified directory. Example: <code>ls /var/log</code>
<code>ls -l</code>	<code>dir</code>	List files in “long” format, one file per line. This also shows you additional info about the file, such as ownership, permissions, date, and size.
<code>ls -a</code>	<code>dir /a</code>	List all files, including “hidden” files. Hidden files are those files that begin with a “.”, e.g. The <code>.bash_history</code> file in your home directory.
<code>ls -ld directory</code>		A “long” list of “directory”, but instead of showing the directory contents, show the directory's detailed information. For example, compare the output of the following two commands: <code>ls -l /usr/bin</code> <code>ls -ld /usr/bin</code>
<code>ls /usr/bin/d*</code>	<code>dir d*.*</code>	List all files whose names begin with the letter “d” in the <code>/usr/bin</code> directory.

.BASH_HISTORY



```
./tst
./test
./test
gcc -std=c99 -o test test.c
./test
gcc -std=c99 -o test test.c
gcc -std=c99 -o test test.c
./test
scp -r ~/Desktop/html/* francesco.santini@ssh.dmi.unipg.it:~/html/
scp -r ~/Desktop/html/* francesco.santini@ssh.dmi.unipg.it:~/html/
scp -r ~/Desktop/html/* francesco.santini@ssh.dmi.unipg.it:~/html/
scp -r ~/Desktop/html/* francesco.santini@ssh.dmi.unipg.it:~/html/
cd Desktop/Programmi/
ls
gcc -std=c99 -o test test.c
gcc -std=c99 -o test test.c
clear
gcc -std=c99 -o test test.c
gcc -std=c99 -o test test.c
gcc -std=c99 -o test test.c
./test
gcc -std=c99 -o test test.c
gcc -std=c99 -o test test.c
gcc -std=c99 -o test test.c
gcc -std=c99 -o test test.c
gcc -std=c99 -o test test.c
:
```

PIPING AND REDIRECTION



PIPING

- ④ The pipe character, “|”, is used to chain two or more commands together.
- ④ The output of the first command is “piped” into the next program, and if there is a second pipe, the output is sent to the third program, etc. For example:
 - ✓ `ls -la /usr/bin | less`
- ④ In this example, we run the command “`ls -la /usr/bin`”, which gives us a long listing of all of the files in `/usr/bin`. Because the output of this command is typically very long, we pipe the output to a program called “`less`”, which displays the output for us one screen at a time.

REDIRECTING OUTPUT TO FILES

- ⌚ There are times when it is useful to save the output of a command to a file, instead of displaying it to the screen.
- ⌚ For example, if we want to create a file that lists all of the MP3 files in a directory, we can do something like this, using the “>” redirection character:

✓ `ls -l /home/vic/MP3/*.mp3 > mp3files.txt`

- ⌚ A similar command can be written so that instead of creating a new file called mp3files.txt, we can append to the end of the original file:

✓ `ls -l /home/vic/extraMP3s/*.mp3 >> mp3files.txt`

OTHER COMMANDS



OTHER COMMANDS

<i>Linux Command</i>	<i>DOS Command</i>	<i>Description</i>
file		Find out what kind of file it is. For example, “ <code>file /bin/ls</code> ” tells us that it is a Linux executable file.
cat	<code>type</code>	Display the contents of a text file on the screen. For example: <code>cat mp3files.txt</code> would display the file we created in the previous section.
head		Display the first few lines of a text file. Example: <code>head /etc/services</code>
tail		Display the last few lines of a text file. Example: <code>tail /etc/services</code>
tail -f		Display the last few lines of a text file, and then output appended data as the file grows (very useful for following log files!). Example: <code>tail -f /var/log/messages</code>
cp	<code>copy</code>	Copies a file from one location to another. Example: <code>cp mp3files.txt /tmp</code> (copies the mp3files.txt file to the /tmp directory)

OTHER COMMANDS

mv	rename, ren, move	Moves a file to a new location, or renames it. For example: <code>mv mp3files.txt /tmp</code> (copy the file to /tmp, and delete it from the original location)
rm	del	Delete a file. Example: <code>rm /tmp/mp3files.txt</code>
mkdir	md	Make Directory. Example: <code>mkdir /tmp/myfiles/</code>
rmdir	rd, rmdir	Remove Directory. Example: <code>rmdir /tmp/myfiles/</code>

```
MacBook-Francesco:~ francescosantini$ rm *.o
```

FIND THINGS

<i>Linux Command</i>	<i>Description</i>
which	Shows the full path of shell commands found in your path. For example, if you want to know exactly where the “grep” command is located on the filesystem, you can type “which grep”. The output should be something like: /bin/grep
whereis	Locates the program, source code, and manual page for a command (if all information is available). For example, to find out where “ls” and its man page are, type: “whereis ls” The output will look something like: ls: /bin/ls /usr/share/man/man1/ls.1.gz
locate	A quick way to search for files anywhere on the filesystem. For example, you can find all files and directories that contain the name “mozilla” by typing: locate mozilla
find	A <i>very</i> powerful command, but sometimes tricky to use. It can be used to search for files matching certain patterns, as well as many other types of searches. A simple example is: find . -name *mp3 This example starts searching in the current directory “.” and all sub-directories, looking for files with “mp3” at the end of their names.

INFORMATIONAL COMMANDS

<i>Linux Command</i>	<i>Explanation</i>
ps	Lists currently running process (programs).
w	Show who is logged on and what they are doing.
id	Print your user-id and group id's
df	Report filesystem disk space usage (“Disk Free” is how I remember it)
du	Disk Usage in a particular directory. “du -s” provides a summary for the current directory.
top	Displays CPU processes in a full-screen GUI. A great way to see the activity on your computer in real-time. Type “Q” to quit.
free	Displays amount of free and used memory in the system.
cat /proc/cpuinfo	Displays information about your CPU.
cat /proc/meminfo	Display lots of information about current memory usage.
uname -a	Prints system information to the screen (kernel version, machine type, etc.)

MORE UTILITIES


<i>Linux Command</i>	<i>Description</i>
clear	Clear the screen
echo	Display text on the screen. Mostly useful when writing shell scripts. For example: <code>echo "Hello World"</code>
more	Display a file, or program output one page at a time. Examples: <code>more mp3files.txt</code> <code>ls -la more</code>
less	An improved replacement for the "more" command. Allows you to scroll backwards as well as forwards.
grep	Search for a pattern in a file or program output. For example, to find out which TCP network port is used by the "nfs" service, you can do this: <code>grep "nfs" /etc/services</code> This looks for any line that contains the string "nfs" in the file "/etc/services" and displays only those lines.

GO FASTER

<i>Shortcut</i>	<i>Description</i>
Up/Down Arrow Keys	Scroll through your most recent commands. You can scroll back to an old command, hit <code>ENTER</code> , and execute the command without having to re-type it.
“history” command	Show your complete command history.
<code>TAB</code> Completion	If you type a partial command or filename that the shell recognizes, you can have it automatically completed for you if you press the <code>TAB</code> key. Try typing the first few characters of your favourite Linux command, then hit <code>TAB</code> a couple of times to see what happens.
Complete recent commands with “!”	Try this: Type “!” followed by the first couple of letters of a recent command and press <code>ENTER</code> ! For example, type: <pre>find /usr/bin -type f -name m*</pre> ...and now type: <pre>!fi</pre>

CHMOD

ls -l



```
-rwxr-xr-x 1 joe acctg 23068 Feb 26 2004 archive.sh
-rw-rw-r-- 1 joe acctg 12878 Jul 24 21:58 orgchart.gif
-rw-rw-r-- 1 joe acctg 2645 Jun 30 08:48 personnel.txt
-rw-r--r-- 1 joe acctg 168 Jul 17 11:51 publicity.html
drwxrwxr-x 2 joe acctg 1024 Mar 18 16:27 sales
-rw-r----- 1 joe acctg 512 Sep 1 07:00 topsecret.inf
-rwxr-xr-x 1 joe acctg 2645 Aug 4 11:03 wordmatic
```

The first character in this column tells what kind of file this is. The dash represents a normal file; the d represents a directory.

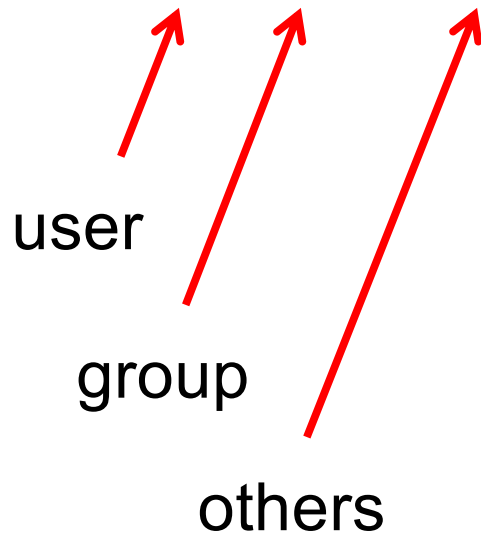
The remaining characters describe the file's **permissions**

The size of the file, in bytes. For directories, this is the size of the directory, not the total of the directory's contents!

PERMISSIONS

```
- rwX  r-X  r-X  
- rw-  rw-  r--  
- rw-  rw-  r--  
- rw-  r--  r--  
d rwX  rwX  r-X  
- rw-  r--  ---  
- rwX  r-X  r-X
```

```
joe  acctg  archive.sh  
joe  acctg  orgchart.gif  
joe  acctg  personnel.txt  
joe  acctg  publicity.html  
joe  acctg  sales  
joe  acctg  topsecret.inf  
joe  acctg  wordomatic
```



CHANGING PERMISSIONS

Before:	-rwx r-x r-x	archive.sh	u = user
Command:	chmod o=r	archive.sh	g = group
After:	-rw xr-x r--	archive.sh	o = others
			a = all
Before:	-rw-r-----	topsecret.inf	r = read
Command:	chmod g=	topsecret.inf	w = write
After:	-rw- --- ---	topsecret.inf	x = execute
Before:	-rw- r-- r--	publicity.html	
Command:	chmod og=rw	publicity.html	
After:	-rw- rw- rw-	publicity.html	

SHELL PROGRAMMING (INTRO)



EXAMPLE

```
#!/bin/bash  
echo Hello World
```

INTRO

- ④ You can put commands in a file and execute them in sequence.

```
#!/bin/bash
# Rotate procmail log files
cd /homes/arensb/Mail
rm procmail.log.6      # This is redundant
mv procmail.log.5 procmail.log.6
mv procmail.log.4 procmail.log.5
mv procmail.log.3 procmail.log.4
mv procmail.log.2 procmail.log.3
mv procmail.log.1 procmail.log.2
mv procmail.log.0 procmail.log.1
mv procmail.log procmail.log.0
```

THINGS TO NOTICE

- ④ The first line of any script must begin with `#!`, followed by the name of the interpreter.
- ④ Comments begin with a hash (`#`) and continue to the end of the line
- ④ A script, like any file that can be run as a command, needs to be executable: save this script (text file) as `rotatelog` and run
 - ✓ `chmod u=rwx rotatelog`
 - ✓ `./ rotatelog`