ITEC202

Operating Systems

Lab Manual

Prepared by

Assoc. Prof. Dr. Ahmet Rizaner

Table of Contents

ITEC 202 – OPERATING SYSTEMS
LABORATORY OUTLINE

- Laboratory sessions are organized in parallel to theoretical study given in classrooms.
- During the lab sessions, particular aspects of the Unix Operating System are demonstrated.
- Students will perform different experiments and submit reports for evaluation each week. Table I is the list of the lab experiments.
- Lab grading is shown in Table II. You must collect at least 50% of the total Lab marks in order to pass the course.
- You must bring a printed copy of the corresponding "Lab Outline" with you to the Laboratory. "Lab Outlines" or "Lab Manual" including all lab experiments have been posted on the course's website. (http://sct.emu.edu.tr/courses/it/itec202)
- Answers of the preliminary Lab Questions must be handled at the beginning of the laboratory.
- You must come to lab at the beginning; any student coming late will be punished by loosing up to 30% of the corresponding lab mark. Students being late more than 20 minutes will not be accepted to the laboratory.
- Eating and drinking in the laboratory is strictly forbidden.
- Before leaving laboratory, you must turn off your computer.
- The following book can be used as an auxiliary book:

  Amir Afraz, *Unix Unbounded*, *A Beginning Approach*, Third Edition, Prentice-Hall, 2000.

Table I. List of Lab Experiments

1	Lab I	MS-DOS Commands
2	Lab II	Interfacing with Unity desktop environment and understanding important UNIX shell commands
3	Lab III	Introduction to the UNIX file system
4	Lab IV	Introduction to Shell Programming
5	Lab V	Shell Programming with Argument Passing
6	Lab VI	Unix Job Control
7	Lab VII	Unix Process Creation

Table 2. Lab Grading

Total Lab Percentage :	16 %
Lab Reports + Performance :	10 % (62.5 % of Lab)
Lab Quiz :	6 % (37.5 % of Lab)

Ubuntu LiveCD

A live CD can be used for a quick demo or test of Ubuntu. By the help of a LiveCD you can try Ubuntu without any changes to your machine. Windows or whatever operating system you use normally is unaffected after trying this and then rebooting. The standard Ubuntu CD can also be used as a Live CD as well as an installer. Live mode is the default option when booting from CD. Live CDs are designed for people that want to use Ubuntu on a computer for a few hours. It is also very useful to practice the ITEC202 laboratory exercises at home.

Preparing your LiveCD

Download Ubuntu from http://www.ubuntu.com/download/desktop. For a live CD, avoid the "alternate CD" & the Server Edition because it has no desktop. Then burn a new cd with the downloaded .iso file.

Using your LiveCD

Put the Ubuntu CD into the CD/DVD-drive and reboot the computer. You should see a menu with "Try Ubuntu" or "Try Ubuntu without any change to your computer". If you don't get this menu, read the booting From the CD guide for more information. Choose "Try Ubuntu" (Fig. 1). You should get a desktop which we call a "LiveCD session". If you don't see a desktop, or need safe graphics mode, read the boot options for more information (Fig. 2)

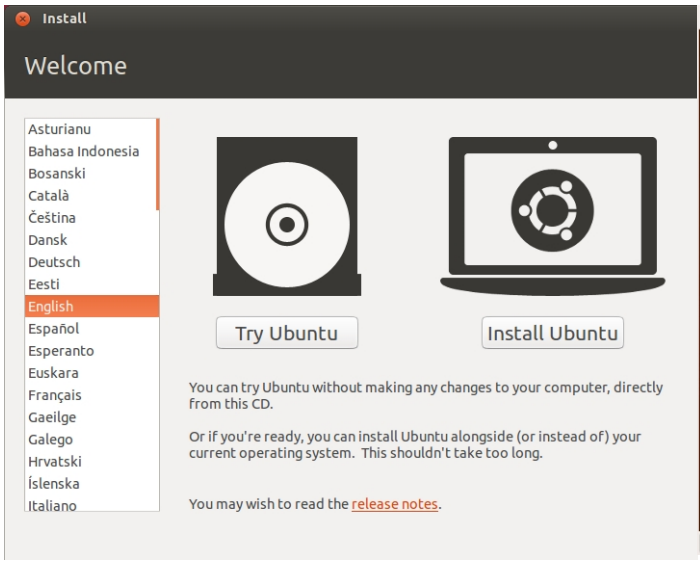


Fig. 1. Try Ubuntu menu

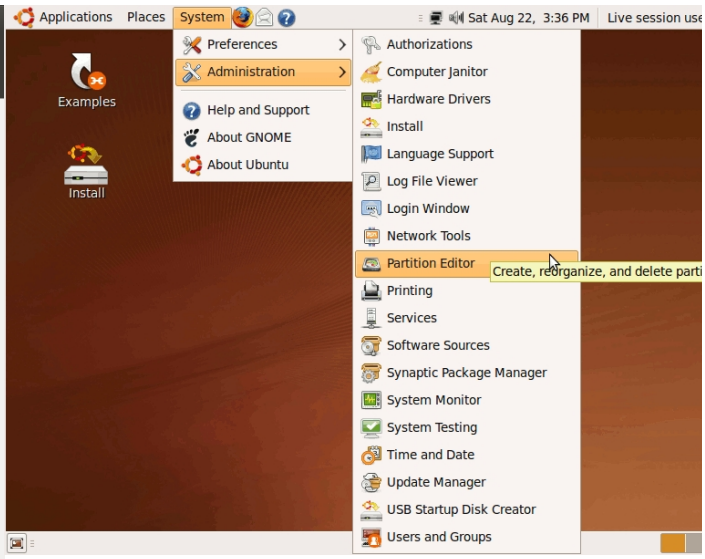


Fig. 2. Ubuntu Desktop

The Firefox icon on the top panel should let you surf the internet. Other normal programs are available in the menus. After you have finished, shut the computer down and remove the CD. At this point anything you saved to the desktop or Documents folders and such will vanish - only things you saved into folders on the hard-drive will remain. This means that there won't be any trace of your personal data (e-mails, passwords etc.) left on the machine.

ITEC202 Operating Systems Lab - Linux/Unix Command Line Cheat Sheet

Command	Description
pwd	prints working directory (prints to screen, ie displays the full path, or your location on the filesystem)
ls	lists contents of current directory
ls –l	lists contents of current directory with extra details
ls *~/*.txt*	lists all files in /home/user ending in .txt
cd	change directory to your home directory
cd ~	change directory to your home directory
cd */scratch/user*	change directory to user on scratch
cd -	change directory to the last directory you were in before changing to wherever you are now
mkdir *mydir*	makes a directory called mydir
rmdir *mydir*	removes directory called mydir. mydir must be empty
touch *myfile*	creates a file called myfile. updates the timestamp on the file if it already exists, without modifying its contents
cp *myfile myfile2*	copies myfile to myfile2. if myfile2 exists, this will overwrite it!
rm *myfile*	removes file called myfile
rm –f *myfile*	removes myfile without asking you for confirmation. useful if using wildcards to remove files ***
cp –r *dir newdir*	copies the whole directory dir to newdir. –r must be specified to copy directory contents recursively
rm –rf *mydir*	this will delete directory mydir along with all its content without asking you for confirmation! ***
nano	opens a text editor. see ribbon at bottom for help. ^x means CTRL-x. this will exit nano
nano *new.txt*	opens nano editing a file called new.txt
cat *new.txt*	displays the contents of new.txt
more *new.txt*	displays the contents of new.txt screen by screen. spacebar to pagedown, q to quit
head *new.txt*	displays first 10 lines of new.txt
tail *new.txt*	displays last 10 lines of new.txt
tail –f *new.txt*	displays the contents of a file as it grows, starting with the last 10 lines. ctrl-c to quit.
mv *myfile newlocdir*	moves myfile into the destination directory newlocdir
mv *myfile newname*	renames file to newname. if a file called newname exists, this will overwrite it!
mv *dir subdir*	moves the directory called dir to the directory called subdir
mv *dir newdirname*	renames directory dir to newdirname
top	displays all the processes running on the machine, and shows available resources
du –h --max-depth=1	run this in your home directory to see how much space you are using. don't exceed 5GB
ssh *servername*	goes to a different server. this could be queso, brie, or provolone
grep *pattern files*	searches for the pattern in files, and displays lines in those files matching the pattern
date	shows the current date and time
anycommand* > *myfile	redirects the output of anycommand writing it to a file called myfile
date > *timestamp*	redirects the output of the date command to a file in the current directory called timestamp
anycommand* >> *myfile	appends the output of anycommand to a file called myfile
date >> *timestamp*	appends the current time and date to a file called timestamp. creates the file if it doesn't exist
command1* | *command2	"pipes" the output of command1 to command2. the pipe is usually shift-backslash key
date | grep *Tue*	displays any line in the output of the date command that matches the pattern Tue. (is it Tuesday?)
tar -zxf *archive.tgz*	this will extract the contents of the archive called archive.tgz. kind of like unzipping a zipfile. ***
tar -zcf *dir.tgz dir*	this creates a compressed archive called dir.tgz that contains all the files and directory structure of dir
time *anycommand*	runs anycommand, timing how long it takes, and displays that time to the screen after completing anycommand
man *anycommand*	gives you help on anycommand
cal -y	free calendar, courtesy unix
CTRL-c	kills whatever process you're currently doing
CTRL-insert	copies selected text to the windows clipboard (n.b. see above, ctrl-c will kill whatever you're doing)
SHIFT-insert	pastes clipboard contents to terminal

*** = use with extreme caution! you can easily delete or overwrite important files with these.

Absolute vs relative paths.

Let's say you are here: /home/turnersd/scripts/. If you wanted to go to /home/turnersd/, you could type: **cd /home/turnersd/**. Or you could use a relative path. **cd ..** (two periods) will take you one directory "up" to the parent directory of the current directory.

.	(a single period) means the current directory
..	(two periods) means the parent directory
~	means your home directory

A few examples	
mv myfile ..	moves myfile to the parent directory
cp myfile ../newname	copies myfile to the parent directory and names the copy newname
cp /home/turnersd/scripts/bstrap.pl .	copies bstrap.pl to "." i.e. to dot, or the current directory you're in
cp myfile ~/subdir/newname	copies myfile to subdir in your home, naming the copy newname
more ../../../myfile	displays screen by screen the content of myfile, which exists 3 directories "up"

Wildcards (use carefully, especially with rm)

*****	matches any character. example: **ls *.pl** lists any file ending with ".pl" ; **rm dataset*** will remove all files beginning with "dataset"
[xyz]	matches any character in the brackets (x, y, or z). example: **cat do[or]m.txt** will display the contents of either doom.txt or dorm.txt

Linux Filesystem

Ubuntu (like all UNIX-like systems) organizes files in a hierarchical tree, where relationships are thought of in teams of children and parent. Directories can contain other directories as well as regular files, which are the "leaves" of the tree. Any element of the tree can be references by a path name; an absolute path name starts with the character **/** (identifying the root directory, which contains all other directories and files), then every child directory that must be traversed to reach the element is listed, each separated by a **/** sign.

A relative path name is one that doesn't start with **/**; in that case, the directory tree is traversed starting from a given point, which changes depending on context, called the current directory. In every directory, there are two special directories called **.** and **..**, which refer respectively to the directory itself, and to its parent directory.

Main directories

The standard Ubuntu directory structure mostly follows the Filesystem Hierarchy Standard, which can be referred to for more detailed information. Here, only the most important directories in the system will be presented.

/bin is a place for most commonly used terminal commands, like ls, mount, rm, etc.

/boot contains files needed to start up the system, including the Linux kernel, a RAM disk image and bootloader configuration files.

/dev contains all device files, which are not regular files but instead refer to various hardware devices on the system, including hard drives.

/etc contains system-global configuration files, which affect the system's behavior for all users.

/home this is the place for users' home directories.

/lib contains very important dynamic libraries and kernel modules

/media is intended as a mount point for external devices, such as hard drives or removable media (floppies, CDs, DVDs).

/mnt is also a place for mount points, but dedicated specifically to "temporarily mounted" devices, such as network filesystems.

/opt can be used to store addition software for your system, which is not handled by the package manager.

/proc is a virtual filesystem that provides a mechanism for kernel to send information to processes.

/root is the superuser's home directory, not in /home/ to allow for booting the system even if /home/ is not available.

/sbin contains important administrative commands that should generally only be employed by the superuser.

/srv can contain data directories of services such as HTTP (/srv/www/) or FTP.

/sys is a virtual filesystem that can be accessed to set or obtain information about the kernel's view of the system.

/tmp is a place for temporary files used by applications.

/usr contains the majority of user utilities and applications, and partly replicates the root directory structure, containing for instance, among others, /usr/bin/ and /usr/lib.

/var is dedicated variable data that potentially changes rapidly; a notable directory it contains is /var/log where system log files are kept.

MS-DOS Command Prompt Window

DOS (Disk Operating System)

DOS stands for disk operating system. The modern computer professional must be familiar with MS-DOS (Microsoft DOS) because MS-DOS remains at the heart of Windows. Although increasingly more configuration can be done through the Windows interface, there comes a time when every computer professional is faced with a screen of MS-DOS commands. Newest versions of Windows operating systems are more independent of MS-DOS than prior versions of Windows. However, previous versions of Windows operating systems (i.e. Windows 98) require autoexec.bat, config.sys and various initialization files. So there remains a need for the computer professional to be familiar with MS-DOS.

MS-DOS Command Prompt

To open a Microsoft DOS command prompt shell window, first click the Start menu at the lower-left of your computer's desktop and select "Run...". Then if you are using Windows XP/Vista/7/8 or 10, type "cmd" (without the quotation marks) into the Run box and click "OK" (Fig. 1).

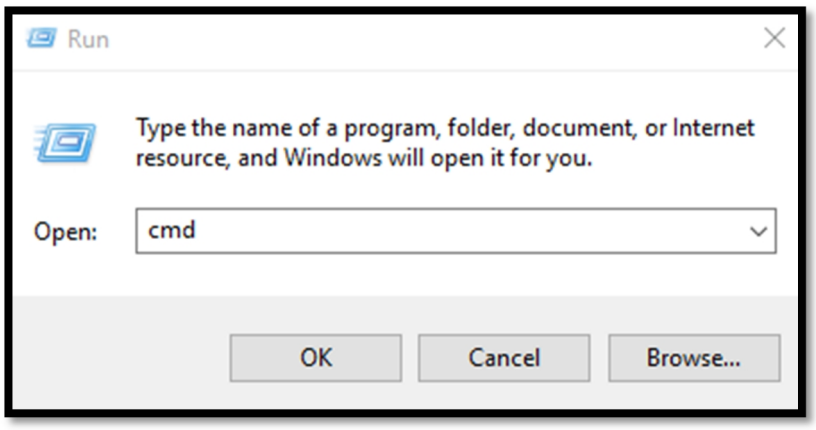


Fig.1. Run box to open MS-DOS command prompt shell window

After you click "OK", a DOS command prompt window will appear. Depending upon which version of Windows you are using, the DOS command window will look similar to Fig.2. To close the window, either type "exit" (without quotations marks) and press Enter, or click the X button in the top-right of the window frame.

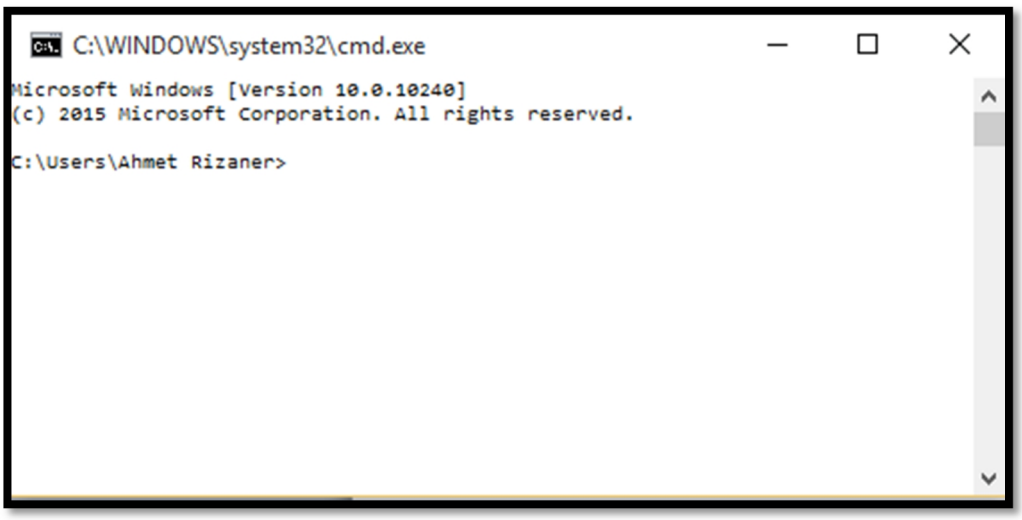


Fig.2. Dos Command Prompt Window

DOS Commands

When you are using a Microsoft DOS command prompt shell window, you can type the following commands into the window.

cd	:	Change directory or display current directory path.
cls	:	Clear the window.
dir	:	Display list of contents of current directory.
help	:	Display list of commands or help about a command.
notepad	:	Run the Windows Notepad text editor.
type	:	Displays the contents of a text file.

Some other useful commands are:

attrib	:	Displays or changes file attributes.
assoc	:	Displays or modifies file extension associations.
chkdsk	:	Checks a disk and displays a status report.
color	:	Sets the text and background colors.
comp	:	Compares the contents of two files or sets of files.
copy	:	Copies one or more files to another location.
date	:	Displays or sets the computer's date
del (or erase)	:	Deletes one or more files.
echo	:	Displays messages, or turns command echoing on/off.
exit	:	Closes the DOS window.
find	:	Searches for a text string in a file or files.
md (or mkdir)	:	Creates a directory.
more	:	Displays the contents of a file one screen at a time.
move	:	Moves one or more files from one directory to another directory.
rd (or rmdir)	:	Removes a directory.
ren (or rename)	:	Renames a file or files.
sort	:	Sorts input.
time	:	Displays or sets the computer's time.
tree	:	Graphically displays the directory structure of a drive or directory.
xcopy	:	Copies files and directory trees.

The following internet related commands are not part of DOS but can be typed at the DOS command prompt:

ftp	:	FTP (file transfer program) to transfer files to/from server.
ipconfig	:	Displays internet configuration, including IP address.
netstat	:	Displays current TCP/IP network connections and statistics.
ping	:	Ping the specified internet IP address or host name.
telnet	:	Starts a text-based telnet session to the specified host.
tftp	:	Transfers files to/from remote computer running TFTP service.
tracert	:	Traces the route to the specified IP address or host.

MS-DOS Commands

1

Part I. In this part basic DOS commands will be introduced.

1. Execute the two MS-DOS commands which will set the default to the root directory (\) on the d: drive.

```
d:
cd \
```

2. Type `date` and `time` and then pres ENTER. DOS will let you check and change the date and time. If the correct date/time is displayed, simply press ENTER. If the date/time is incorrect, type the correct date/time and press ENTER.
3. Type `ver` to display the Windows OS's version number.
4. Use the `dir` command to list the contents of the root directory on the d: drive (i.e. `d:\> dir`).
5. Type "`help` [`command`]" to display help information on that command. (i.e. `help dir`)

Part II. In the steps below we will create a simple text file called hello.txt, make two directories, mydir and mysubdir and copy hello.txt into the subdirectories. We will then delete the copies of hello.txt and remove the directories under mydir.

1. Invoke the MS-DOS full screen text editor utility EDIT (if your OS does not support EDIT then use notepad command i.e. `notepad hello.txt`) to create a text file called hello.txt:

   ```
   d:\> notepad hello.txt
   ```

   Enter the line "Hello World!" use File|Save and File|Exit to save your work and to quit EDIT.

2. Use the `type` command to display contents of hello.txt.

   ```
   d:\> type hello.txt
   ```

   You should see: Hello World!

3. Use the `md` command to create (make) a new directory called mydir.

   ```
   d:\> md mydir
   ```

4. Use the `cd` command to change the default directory to mydir.

   ```
   d:\> cd mydir
   ```

5. Use dir to display the contents of mydir. It should contain no files except for "pointers" to itself and its parent.

   ```
   d:\mydir> dir
   ```

6. Use the `copy` command to copy the file hello.txt in the root directory to mydir.

   ```
   d:\mydir> copy d:\hello.txt
   ```

7. Use `dir` to display the contents of mydir. You should see hello.txt listed.

   ```
   d:\mydir> dir
   ```

8. Use `md` to create (make) a subdirectory for mydir called mysubdir.

   ```
   d:\mydir> md mysubdir
   ```

9. Use `cd` to change the default directory to mysubdir.

   ```
   d:\mydir> cd mysubdir
   ```

10. Copy hello.txt to mysubdir. Use the mydir directory copy of hello.txt.

```
d:\mydir\mysubdir> copy d:\mydir\hello.txt
```

11. Make a second copy of hello.txt but call it hello1.txt.

```
d:\mydir\mysubdir> copy d:\hello.txt hello1.txt
```

12. Use the `ren (rename)` command to rename hello.txt as hello2.txt.

```
d:\mydir\mysubdir> rename hello.txt hello2.txt
```

13. Check your work - display the contents of mydir.

```
d:\mydir\mysubdir> tree \mydir
```

14. Check that the content of hello2.txt has not changed by displaying it.

```
d:\mydir\mysubdir> type hello2.txt
```

15. Using a wildcard delete all files with a .txt extension in mysubdir.

```
d:\mydir\mysubdir> del *.txt
```

16. Use the `dir` command to check that both files are gone.

```
d:\mydir\mysubdir> dir
```

17. Return to the mydir directory.

```
d:\mydir\mysubdir> cd \mydir
```

18. Use the `dir` command to view the contents of the mydir directory.

```
d:\mydir> dir
```

19. Use the `rd` command to remove the mysubdir directory.

```
d:\mydir> rd mysubdir
```

20. Use the dir command to check that the mysubdir subdirectory is gone.

```
d:\mydir> dir
```

21. Return to the root directory.

```
d:\mydir> cd \
```

Part III. Useful DOS Commands

1. **ipconfig**: Displays the current IP address of your computer and the DNS server address. Try and observe the outputs of "`ipconfig`" and "`ipconfig /all`".

   What are the IP and MAC (Media Access Control) addresses of your computer?

2. **assoc**: Displays the file associations. For example the command "`assoc .avi`" will quickly tell you the name of your default movie player.

   Try "`assoc .doc`".

3. **ping**: Ping network command followed by the web-address or IP address tells you whether the other party is responding to your handshake request. Ping tool can also be used to convert the web address to a physical IP address.

   Try "`ping google.com`".

4. **attrib**: Attrib lets you to change attributes of system files and even hidden files.

   Type "`attrib +h hello.txt`" to make hello.txt a hidden file.

   Type "`dir`" to see the list of the files and "`dir /a`" to see the list of the hidden files as well.

   Make hello.txt visible again.

ITEC 202 – Preliminary Lab Questions – Lab 1		
	Student No. :	Mark
	Name Surname :	

Carefully read the description of the corresponding lab. The lab outline contains background for the lab and directions for doing the lab procedure. There may also be handouts or other materials you have access to. Answers of the preliminary Lab Questions must be handled at the beginning of the laboratory. A full printable version of this sheet is available online.

1. What is the overall purpose of the lab?

2. What MS-DOS command is used to remove or delete a directory?

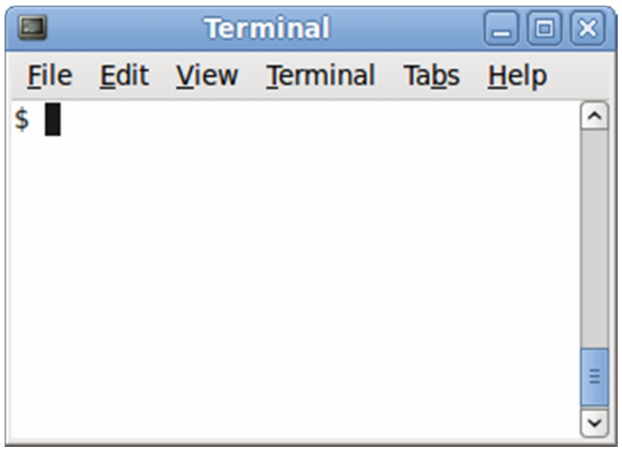
3. What is the function of "ipconfig" command?

4. What is the function of "tree" command?

5. What command is used to list all files including the hidden files under the current working directory?

Part I. Unity desktop environment

1. Unity is the default desktop environment for Ubuntu 12.04 (Precise Pangolin) and later. It enables users to easily use and configure their computers.



2. The easiest way to open the Terminal is to use the 'search' function on the dash. Or from the Unity desktop, in the upper left corner, go to **Dash -> More Apps -> Accessories -> Terminal**. The black window pop up with a $ prompt is called the command line window.
3. When you start Terminal for the first time, the application opens a terminal window with a group of default settings. The group of default settings is called the Default profile.
4. Enter the command **lsb_release -a**. Your version will be shown on the Description line.
5. Type into terminal: **unity --version**. The Unity version will be shown.
6. Type : **uname**. The name of the operating system will be shown.

Part II. Getting help about UNIX commands by using man command.

1. Use shell command **man** to get and read the descriptions of the shell commands, **who**, **whoami**, **id**, **cat**, **ls**, **cd**, **pwd**, **mkdir**. (*i.e*. **$ man ls**)

Part III. Some basic UNIX commands such as date and time.

1. Use **date** command to see the current date and time.
2. Use **cal** command to display the calendar for the current month.
3. Display the calendar for November 2010. Hint: use **man cal** command to get help on cal command.
4. Use **cal** command to look at the calendar for the year 2011.

Part IV. A simple text editor pico for creating some text files.

1. Use **pico** command to start text editor.
2. Create several text files named ayse.txt, ali.txt, jane.log, lena.log, loop.co and sky.to. You are free to write any text into these files.

3. Use **ls** command to see if all of the files mentioned in the above step are created, if not, create the missing files.
4. Use **pico** to edit jane.log (open jane.log and add some more text into this file), i.e. **pico jane.log**.
5. Please check if all files listed in part 2 are created. You can also use gnome default text editor **gedit** to create the missing files.

Part V. ls (list) command.

1. Type **pwd** (print working directory) command to learn your current directory.
2. Type **ls** command to see the list of the files in your current directory.
3. Type **ls –a** command to see the list of the hidden files i.e., the files starting with dot.
4. Type **ls –l** command to list files in a long format, showing detailed information about the files.
5. Type **ls -1** command to list one file per line.
6. Use **ls** command to list only the files starting with letter a. Hint: user wildcards ***** is used in place of strings of characters (*i.e.* **bb*** means all the files starting with bb and **?** is used only for one character (*i.e.* **a?bc** means all the 4 characters long files starting with a and ending with bc.)
7. Use **ls** command to list all the files having an extension of log.
8. Use **ls** command to list all the files whose third letter of the filenames is n.
9. Use **ls** command to display all the files whose first letter is l and third letter is n.

Part VI. pipes, redirection, cat and wc commands.

1. **cat** command is used to display the content of a text file on the screen. Use **cat ayse.txt** to display the contents of ayse.txt.
2. Display the contents of two files at the same time. *i.e*. **cat ayse.txt ali.txt**.
3. Redirection (**>**) symbol to redirect the output of a command. For example, to create a file that contains the list of files, type **ls -1 > filelist**.
4. Tye **cat filelist** to display the content of filelist.
5. **wc** (word count) command counts the number of characters, words, or lines in a specified file. Type **wc –l filelist** to count number of lines (files) in filelist.
6. Use **wc** command to count number of words in userlist. *i.e*. **wc –w filelist**.
7. The pipe (**|**) makes the output of one command the input of another command. To get a sorted list of files type: **ls -1 | sort** (same as **sort filelist**).

ITEC 202 – Preliminary Lab Questions – Lab 2		
	Student No. :	Mark
	Name Surname :	

Carefully read the description of the corresponding lab. The lab outline contains background for the lab and directions for doing the lab procedure. There may also be handouts or other materials you have access to. By the help of a LiveCD of Ubuntu you can study to the corresponding lab without making any changes to your machine. Answers of the preliminary Lab Questions must be handled at the beginning of the laboratory. A full printable version of this sheet is available online.

1. What is the overall purpose of this lab?

2. What is the function of "whoami" command?

3. What is the function of "ls" command?

4. What is the name of the default desktop environment of Ubuntu 12.04?

5. Write the necessary commands to display the list of the files ending with txt under the current working directory.

Introduction to the UNIX file system

3

Part I. The `mkdir` (make directory) command creates a new subdirectory under your working directory or any other directory you specify as a part of the command.

1. Use **`pwd`** command to check your working directory. Output of **`pwd`** should be similar to: /home/username
2. Create a directory named memos. i.e. type **`mkdir memos`**.
3. Type **`ls`** and check if directory named memos is created.

Part II. The `cd` (change directory) command is used to change directory.

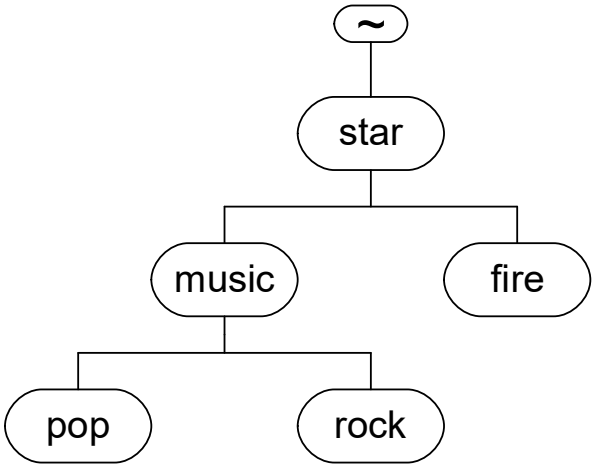
1. Type **`cd`** command. This command will take you to your home directory.
2. Type **`pwd`** to check if you are in your home directory. i.e. /home/username
3. Type **`cd memos`** to change your working directory to memos, and then check your working directory by **`pwd`** command. Output of **`pwd`** should be /home/username/memos
4. Create another directory called box, and change your directory to box. Again check your working directory with **`pwd`**. Output of **`pwd`** should be /home/username/memos/box
5. Type **`cd ..`** this command will take you one directory up. Check if your working directory is memos.
6. Repeat 1 to return your home directory. i.e. type **`cd`**.

Part III. In this part you will learn how to copy, move and erase files. `cp` (copy) command is used for copying files, `mv` (move) command is used to move files and `rm` (remove) command is used to erase files. In this section it is assumed that in your home directory these files are present ayse.txt, ali.txt, lena.log, sky.to and loop.co, if not create these files using `pico` or `gedit` editor.

1. Use **`man`** command to get help on these commands: **`cp`**, **`mv`** and **`rm`**.
2. Copy lena.log into memos directory. i.e. **`cp lena.log memos`**.
3. Change your directory to memos and check if lena.log is there. i.e. use ls.
4. Remove lena.log by using **`rm`** command. i.e. **`rm lena.log`**. Then check if it is erased.
5. **`mv`** command can also be used to rename filenames. i.e. **`mv ayse.txt veli.dat`** will rename ayse.txt as veli.dat.
6. Type **`cp veli.dat elena.rtr`**. What is the difference of this command from the command investigated in step 5.

Part IV. In this part following directory structure will be created and tree command will be investigated.

1. Type **`tree`** command to see directory structure under your home directory. This command also displays the number of directories and files. i.e. **`tree ~`**.
2. Use **`mkdir`** command to create the directory structure shown below.



3. Return to your home directory and type **`tree`** to see if the shown directories are created correctly: type **`cd`** and then **`tree`**. ***Show your work to the instructor***.
4. Return to home directory and try to change your working directory to pop directory by writing a single command. i.e. **`cd star/music/pop`**. And confirm by **`pwd`** command.
5. Copy all the files starting with character a to fire directory. i.e. **`cp ~/a*.* ~/star/fire`**. Please note that the tilde (**`~`**) used before slash (**`/`**) means home directory.
6. Change your working directory to fire and remove all the files inside this directory.
7. Remove fire directory as well. i.e. you should use **`rmdir`** command.
8. The **`-p`** (i.e., parents) option creates the specified intermediate directories for a new directory if they do not already exist. For example, it can be used to create the following directory structure: **`mkdir -p food/fruit/citrus/oranges`**.
9. Type **`tree`** to check what directory structure has been created in step 8.

Part V. The `rmdir` (remove directory) command is used to remove directory.

1. Use **`rmdir`** command to remove all the directories under the star directory. Hint: start form the inner directories (i.e. pop), the directory that you wish to remove should be empty. ***Show that you remove all the directories correctly to the instructor***.

Note: The commands learned in this lab section will be used in the subsequent laboratory work.

ITEC 202 – Preliminary Lab Questions – Lab 3		
	Student No. :	Mark
	Name Surname :	

Carefully read the description of the corresponding lab. The lab outline contains background for the lab and directions for doing the lab procedure. There may also be handouts or other materials you have access to. By the help of a LiveCD of Ubuntu you can study to the corresponding lab without making any changes to your machine. Answers of the preliminary Lab Questions must be handled at the beginning of the laboratory. A full printable version of this sheet is available online.

1. What is the overall purpose of this lab?

2. What is the meaning of the following command: "mv file1.txt file2.dat"

3. What is the function of "rmdir" command?

4. What Unix command is used to create a new directory?

5. What symbol can be used to represent the users home directory?

Introduction to Shell Programming

4

Part I. Changing file mode. Only the owner of a file can use `chmod` (change file mode) to change the permission of a file.

Symbol	Meaning	Symbol	Meaning	Symbol	Meaning
u	user	**a**	all	**x**	execute (and access directory)
g	group	**r**	read	**+**	add permission
o	other	**w**	write (and delete)	**-**	take away permission

1. Create a text file by redirecting the output of **`ls -1`** command into a text file called files.txt. i.e. **`ls -1 > files.txt`** .
2. Type **`ls –l files.txt`** to check the permissions of this file. (Permissions consist of 10 characters i.e. **`-rw-r--r--`**. The first character indicates the file type and the rest indicate the file access modes for user, group and all the others respectively)
3. Try the following commands and then observe the difference in the permissions again typing **`ls –l files.txt`** as in 2.

   `chmod a-rw files.txt`

   `chmod go+rw files.txt`

Part II. Displaying information: The `echo` command can be used to display messages. It displays its arguments on your terminal.

1. Type **`echo “This is a test”`** and observe the output.
2. Type this command, **`echo “Total number of files is :” `ls -1 | wc –l``** .
3. **`echo`** command can be used to display text and the values of the shell variables. To access the value stored in a shell variable, you must precede the name of the variable with a **`$`**. Set age to 25. i.e. **`age=25 or age=”25”`** .
4. Try the following commands and observe the differences:

   `echo “You are age years old”`

   `echo “You are $age years old”`
5. Observe the outputs of the following commands.

   `echo $HOME`

   `echo ~`

Part III. Simple shell script.

1. Write a simple shell script to display number of users in the system by using **`pico`** utility as shown below and save it as ***won***.

```
# won
# displays # of files and directories
echo "Number of files & directories :"
ls -1 | wc -l
```

2. You can use **sh** command to execute script files. To run won script, type **sh won**.

Part IV. Executable shell scripts by `chmod` command.

1. The second method of executing a shell program is to make the script file executable. In this case, all you do is to type the name of the script program; just as you do for any other shell commands. This is preferred method. Try the following set of commands:

1	**chmod u+x won**	Makes ***won*** script an executable file for the user
2	**./won**	Execute ***won*** since it is an executable file

Part V. The cases construct.

1. The shell provides you with the case construct, which enables you to selectively execute a set of commands from the list of commands. An example is given below; use **pico** utility to write the following shell script. Script name should be MENU.

```
# MENU
echo " 0: Exit"
echo " 1: Show Date and Time"
echo " 2: List my HOME directory"
echo " 3: Display Calendar"
echo " Enter your choice:"
read option
case $option in
    0) echo "good bye" ;;
    1) date ;;
    2) ls $HOME ;;
    3) cal ;;
    *) echo "Invalid input. Good bye." ;;
esac
```

2. Make **MENU** an executable script. i.e. **chmod u+x MENU**
3. Type **./MENU** and try to understand the shell script that you have written.

Note: The commands learned in this lab section will be used in the subsequent laboratory work.

ITEC 202 – Preliminary Lab Questions – Lab 4		
	Student No. :	Mark
	Name Surname :	

Carefully read the description of the corresponding lab. The lab outline contains background for the lab and directions for doing the lab procedure. There may also be handouts or other materials you have access to. By the help of a LiveCD of Ubuntu you can study to the corresponding lab without making any changes to your machine. Answers of the preliminary Lab Questions must be handled at the beginning of the laboratory. A full printable version of this sheet is available online.

1. What is the overall purpose of this lab?

2. What is the meaning of the following command: "chmod go+rw files.txt"

3. What is the function of "sh" command?

4. Write the command necessary to make script file "xyz" an executable file for the user?

5. What is the output of the following command: "echo ~"?

Shell Programming with Argument Passing

5

Part I: Passing arguments to a shell script.

1. Create the below script and save it as **example1**. Make it an executable file (**chmod u+x example1**), then run and observe the output (**./example1 david jane lena**).

```
# Passing arguments to a shell script
echo "1. argument: $1"
echo "2. argument: $2"
echo "3. argument: $3"
echo "Total number of arguments: $#"
echo "All arguments: $*"
```

Part II: IF expression.

1. Create the bellow script and save it in a file named **example2**. Then use **chmod** command to make **example2** executable and run it to observe the output. (**./example2**).

```
# Input from the keyboard
echo "Do you want to exit (y/n)?"
read ans
if [ "$ans" = y ]
      then
      echo "Exiting..."
      exit 0
fi
```

Part III: CASE statement with arguments.

1. Create an executable shell program shown below named **example3**. Then run **example3** and observe the output. (./**example3 list**).

```
# Sample shell program with CASE
# Execute given command
echo "The command entered is: '$1'"
case "$1" in
   whoami)
         echo "The command whoami is being executed..."
         whoami
         ;;
   list)
         echo "The command ls is being executed..."
         ls
         ;;
   cal)
```

```
          echo "The command cal is being executed..."
          cal
          ;;
     *)
          echo "Wrong command, valid commands: whoami, list, or cal"
          ;;
  esac
```

Part IV. A Script to rename filename where old and new file names are passed as arguments.

1. Create the below shell script, name it as **rnfile**, make it executable and then run to observe the output. (./**rnfile ali.txt veli.log**).

```
# rnfile
# File renaming script

clear
if [ $# == 2 ]; then              # check if two argument were entered
    if [ -f $1 ]; then            # check whether the file exists or not
      mv $1 $2

      if [ -f $2 ]; then              # check if the new file exists
        echo "Renaming"
        echo "-------------------------"
        echo "$1 is renamed as $2."

        exit 0
      else                            # if new file cannot be created
        echo "$1 cannot be renamed as $2!"
        exit 0

      fi

      exit 0

    else                                # if file does not exist
     echo "$1 does not exist."
     exit 1
    fi

  exit 0
  else                                # if two arguments were not entered
    echo "Two arguments must be entered.."
    echo "example: ./rnfile file1 file2"
  exit 1
fi
```

Note: The commands learned in this lab section will be used in the subsequent laboratory work.

ITEC 202 – Preliminary Lab Questions – Lab 5		
	Student No. :	Mark
	Name Surname :	

Carefully read the description of the corresponding lab. The lab outline contains background for the lab and directions for doing the lab procedure. There may also be handouts or other materials you have access to. By the help of a LiveCD of Ubuntu you can study to the corresponding lab without making any changes to your machine. Answers of the preliminary Lab Questions must be handled at the beginning of the laboratory. A full printable version of this sheet is available online.

1. What is the overall purpose of this lab?

2. Write a shell script to prompt the user to input her/his age and print it on the screen only if the entered age is less than 30 as shown below:

Your age is 20 years old.

Part I. Displaying the PID (process id): The `ps` command.

1. A process is an executing program identified by a unique PID (process identifier). To see information about your processes, with their associated PID and status type: **`ps`** and **`ps –l`**
    - **PID:** The process id, a unique identifier assigned to each process
    - **UID:** The user who owns the process
    - **TTY:** Your terminal number that control the process
    - **TIME:** Time duration in second that your process is running
    - **CMD:** The command used to invoke the process

Part II. Timing a delay: The `sleep` command. The sleep command causes the process executing it to go to sleep for a specified number of second.

1. Type this command: `sleep 15; echo "I am awake."`

   The `sleep` command executed causing 15 seconds delay, and then the `echo` command is executed.

Part III. Background processing: Using the Ampersand (&). UNIX is a multitasking system; it allows you to execute several programs concurrently. Usually you type a command and within a few second output of the command is displayed. What if you run a command that takes minutes to execute? In this case, you have to wait for the command to finish executing before you can proceed with the next job. However, you can run long programs in the background.

1. Type the following commands:

   a. `ls > temp&`

   This command runs the `ls` command in the background, with its standard output redirected to a file named temp. As the shell processes the command, it will display a job number and a processed, i.e. `[1] 6167`.

   b. `sleep 10&`

   The & at the end of the command line runs the job in the background and returns the prompt straight away, allowing you to run other programs while waiting for that one to finish. The job number and PID will be returned by the machine. The background command process ID number can be used to terminate the process or obtain its status.

   c. `(sleep 120; echo "I am awake")&`

   Above command runs in the background.

d. `ps`

You can try any other commands to investigate the multitasking ability of UNIX operating system.

Part IV. Backgrounding a current foreground process

1. At the prompt, type: `cat > trial.txt`
2. You can suspend the process running in the foreground by typing (control+z): ^Z
3. Then to put it in the background, type: `bg`
4. Type: `sleep 2000` and then ^Z and `bg` to run the processes in background.
5. To list all jobs that are running from the current shell, the `jobs` command may be used. Type: `jobs`

Part V. Restarting a background processes: The `fg` command

1. Jobs command will return you the job numbers. To restart (foreground) a suspended processes, type: `fg %job-number`. Check the job number of "cat > trial.txt". Assuming that it is 2, the following command will bring that job foreground: `fg %2`
2. Then type the following to quit that process. (control + d): ^D
3. Typing `fg` with no job number foregrounds the last suspended process.

Part VI. Killing a background processes: The `kill` command

1. It is sometimes necessary to kill a process (for example, when an executing program is in an infinite loop). To kill a job running in the foreground, type ^C (control + c). For example, Type:

   ```
   sleep 100
   ^C
   ```

2. To kill a suspended or background process, type: `kill %job-number`
3. For example, type the following commands:

   ```
   sleep 100 &
   jobs
   ```

4. If it (sleep 100) is job number 4, type `kill %4`
5. Alternatively, processes can be killed by finding their process numbers (PIDs, by ps command) and using `kill PID_number`. Assume that the PID number of sleep 2000 command is 6377. To kill this process type:

   ```
   kill 6377
   ```

6. Kill all the processes running in background.

Note: The commands learned in this lab section will be used in the subsequent laboratory work.

ITEC 202 – Preliminary Lab Questions – Lab 6		
	Student No. :	Mark
	Name Surname :	

Carefully read the description of the corresponding lab. The lab outline contains background for the lab and directions for doing the lab procedure. There may also be handouts or other materials you have access to. By the help of a LiveCD of Ubuntu you can study to the corresponding lab without making any changes to your machine. Answers of the preliminary Lab Questions must be handled at the beginning of the laboratory. A full printable version of this sheet is available online.

1. What is the overall purpose of this lab?

Assume that the followings are the outputs of ps and jobs commands and answer the following questions according to these outputs.

ps

```
  PID TTY          TIME CMD
 6417 pts/0    00:00:00 bash
 6458 pts/0    00:00:00 sleep
 6459 pts/0    00:00:00 bash
 6460 pts/0    00:00:00 sleep
 6468 pts/0    00:00:00 cat
 6469 pts/0    00:00:00 ps
```

jobs

```
[1]   Running  sleep 5000 &
[2]-  Running  (sleep 250; echo "Hello") &
[3]+  Stopped   cat >list.txt
```

2. What are the process ID no, job number and status of the command "sleep 5000"

   Process ID: _____ Job no: _____ Status: _____

3. Write the necessary command line to restart (foreground) the third job in the job list.

Part I. Using the C Language.

C is a general purpose language which has features of a high-level procedural language, and yet allows some very low-level operations as well. Many operating systems are written in C. In particular, most of the UNIX operating system utilities are written in C, so we will be using this language when we wish to experiment with UNIX and its utilities.

1. Use `pico` to type and save the following program as lab1.c

```
/*****************************
  Lab 1, Exercise 1
*****************************/
#include <stdio.h>
int main() {
  printf("Rob McArthur\nHello\n");
  return 0;
}
```

2. To compile a program we use the `gcc` command. To compile above program type:

```
gcc –o lab1 lab1.c
```

3. The above command says to compile the C source file called `lab1.c`, and put the executable program into a file called lab1 (`–o lab1`).
4. To run the program just type:

```
./lab1
```

`./` means that the file is in the current directory. To avoid using `./` before the command, you can move your executable files into `~/bin` directory.

Part II. Child and Parent processes.

1. The following unix calls are used in this lab: `getpid`, `getppid`, `sleep` and `fork`. Use the on-line help (`man getpid`) for further information on Unix system calls.

2. Use `pico` to type and save the following program as `lab2.c`. Compile and run the program twice, both times as a background process (type: `./lab2&` twice). Once both processes are running as background processes, view the process table (use `ps -l`) and observe the process identification numbers. If you see the message "`I am awake`", the process has finished and will no longer show up in the process table.

```
#include <stdio.h>
main()
 {
  printf("Process ID is: %d\n", getpid());
  printf("Parent process ID is: %d\n", getppid());
  sleep(120); /* sleeps for 2 minutes */
  printf("I am awake.\n");
 }
```

3. Compile the following program as `lab3.c`. Run the `lab3` and observe the number of statements printed to the terminal. The `fork()` call creates a child that is a copy of the parent process. Because the child is a copy, both it and its parent process begin from just after the `fork()`. All of the statements after a call to `fork()` are executed by both the parent and child processes. However, both processes are now separate and changes in one do not affect the other.

```
#include <stdio.h>
main()
{
  fork();
  fork();
   printf("Process ID is: %d\n", getpid());
   printf("Parent Process ID is %d\n", getppid());
   sleep(2);
}
```

4. Compile and run the following program as `lab4` and observe the output. In Unix, `fork()` returns the value of the child's PID to the parent process and zero to the child process.

```
#include <stdio.h>
main()
{
  int ret;
  ret = fork();
  if (ret == 0)
  {
   /* this is the child process */
   printf("The child process ID is %d\n", getpid());
   printf("The child's parent process ID is %d\n", getppid());
  }
  else
  {
  /* this is the parent process */
    printf("The parent process ID is %d\n", getpid());
    printf("The parent's parent process ID is %d\n", getppid());
  }
  sleep(2);
}
```

ITEC 202 – Preliminary Lab Questions – Lab 7		
	Student No. :	Mark
	Name Surname :	

Carefully read the description of the corresponding lab. The lab outline contains background for the lab and directions for doing the lab procedure. There may also be handouts or other materials you have access to. By the help of a LiveCD of Ubuntu you can study to the corresponding lab without making any changes to your machine. Answers of the preliminary Lab Questions must be handled at the beginning of the laboratory. A full printable version of this sheet is available online.

1. What is the overall purpose of this lab?

2. What is the function of "getpid" ?

3. Write the full command line to compile the C source file called map.c, and put the executable program into a file called map.

4. What is the function of "gcc" command?

5. What is the name of the system call that creates a child process that is a copy of the parent process?

Additional Notes