Lab 1 (OS): Introduction to the UNIX shell & the C language

Shokhista Ergasheva, Muwaffaq Imam, Artem Kruglov,
Nikita Lozhnikov, Giancarlo Succi, Xavier Vasquez,
Herman Tarasau, Firas Jolha

Week 01 – Lab

Lab Objectives

1. Install a Linux-based operating system.
2. Learn some basic shell commands.
3. Learn how to compile and run a simple program in C language.

Outline

- Installing a Virtual OS
- The UNIX Shell
- The C Language

Installing a Virtual Operating System (1/2)

Virtual operating system[1] is an operating system that runs on a virtual machine.

Virtual machine is an emulation of a computer system comprised of specialized software. A virtual computer has its own RAM, hard drive, processor and so forth. Basically, this is a separate computer in a physical computer which runs on shared physical hardware resources (RAM, processor and hard drive).

The virtual OS is called the *guest OS*, whereas the primary OS is the *host OS*.

[1] https://book.cyberyozh.com/virtual-machine-and-virtual-operating-system/ © CyberYozh security group

Installing a Virtual Operating System (2/2)

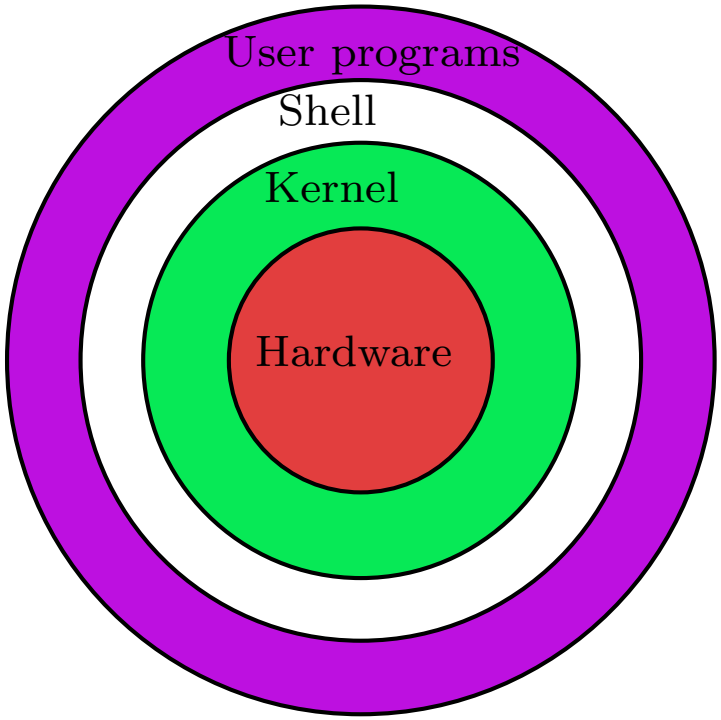
Installing Ubuntu (Linux distribution) on a virtual machine using VirtualBox.

- Download VirtualBox from https://www.virtualbox.org/wiki/Downloads
- Download the ISO image of Ubuntu 22.04.1 (Jammy Jellyfish) from the official website (https://ubuntu.com/download/desktop).
- Install Ubuntu by following the tutorial.

Note: If you have Linux based operating system installed, then you can skip this slide.

What is the UNIX shell?

- **Shell** is a text user interface (TUI) for access to an operating system's services. Has many implementations: bash shell, original Unix shell, Bourne shell, ksh, csh, etc.



Architecture of UNIX Systems

What is the UNIX shell?

- The job of the **Shell** is to translate the user's command lines into operating system instructions.

What is the UNIX shell?

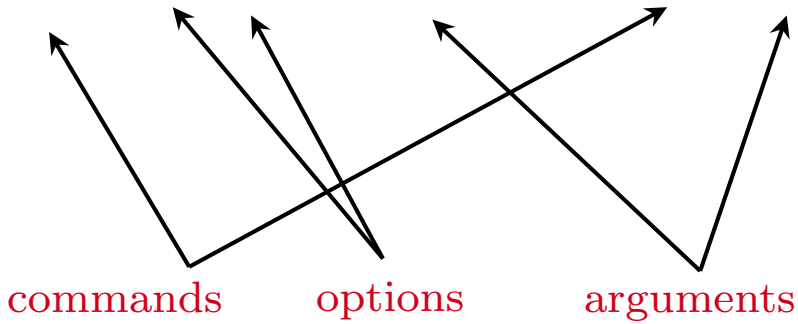
- The job of the **Shell** is to translate the user's command lines into operating system instructions.
- For example, consider this command line:

```
sort -n -r namelist > namelist.sorted
```

What is the UNIX shell?

- The job of the **Shell** is to translate the user's command lines into operating system instructions.
- For example, consider this command line:

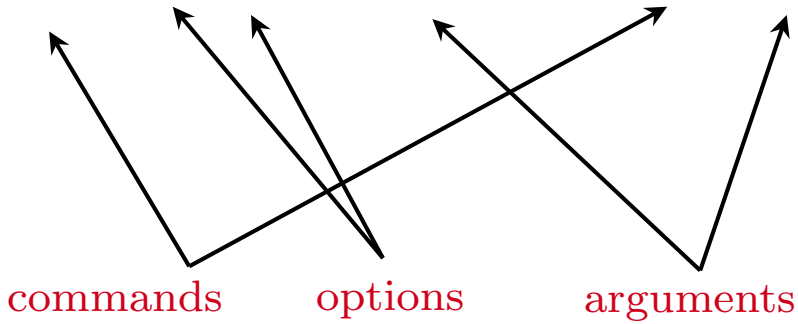
```
sort -n -r namelist > namelist.sorted
```



What is the UNIX shell?

- The job of the **Shell** is to translate the user's command lines into operating system instructions.
- For example, consider this command line:

```
sort -n -r namelist > namelist.sorted
```



This means, "Sort lines in the file *namelist* in numerical and reverse order, and put the result in the file *namelist.sorted*."

What is the UNIX shell?

Example 1:

- create a file *namelist* and add some names.
- sort the names in reversed order using shell command **sort**.
- store the sorted names in another file *namelist.sorted*

- **whoami** - Print userid.
- **hostname** - Show the system's host name.
- man << *item* > - Display manual for the < *item* >. Use arrows to navigate and q to exit. Example: **man whoami** - Display manual on command **whoami**.
- **man man** - Display man on man.
- **man --help** - The other way to get help on command is to write an option **--help** or often **-h**.

Shell - Display

- **less** - Display the contents of a file one screen at a time with navigation.
- **head** - Print the first lines of the file to standard output.
- **tail** - Print the last lines of the file to standard output.
- **man -h — head**
- **man –help — tail**
- **grep PATTERN** $< file >$ - Search for PATTERN in file or stdin.

Shell - Streams

Standard streams are preconnected communication channels of programs. They are:

- **stdin** - standard input that going into program,
- **stdout** - standard out where program writes output,
- **stderr** - to display error messages.

It is possible to redirect streams to or from files with $>$ and $<$.

Shell - Pipelines

- **ls > list.txt** - Save list of files in current directory to file.txt.
- **head -n 3 < file.txt** - Display the first 3 entries.
  - **file.txt** is an input stream for **head** command.

  It is possible to redirect output of one program to input of another by | (pipe symbol).
- **ls | sort -r | tail -n 3**
  Get list of files, reverse sort and display the 3 last.

Streams and Pipelines

Example 2:

- search for file names which contain the word "**driver**" in the path "**/proc**".
- save the output of the previous command in a file *output.txt* and the errors in another file *errors.txt*.
- print the first 3 lines from the files *output.txt* and *errors.txt* in the shell after sorting them in alphabetical order (Hint: Use the input streams).

Shell - File system commands

- **pwd** - Print name of current/working directory.
- **mkdir <dirname>** - Make directory.
- **cd <path>** - Change directory.
- **rm <filenames>** - Remove a file.
- **rm -r <dirname>** - Remove (recursive) a directory.
- **ls** - List content of a directory.
- **mv <old_path> <new_path>** - Move file.
- **cat <filenames>** - Concatenate files to stdout.
- **gedit <filename>** - Run text editor for GNOME.

Shell - File System - Special Characters

- ~ - home directory
- . - represent current directory
- .. - represent parent directory of current directory
- Examples:
  - cd ..
  - ls .
  - cd ~

Shell - File System FAQ

- **Q: How to create a new file?**
  touch <filename>
  cat > <filename>
  echo > <filename>
  gedit <filename>
- **Q: How to rename file?**
  mv <oldname> <newname>

Foreground and Background

Foreground processes block shell during execution whereas **background** processes do not. Appending & will run process in background.

- **gedit &**

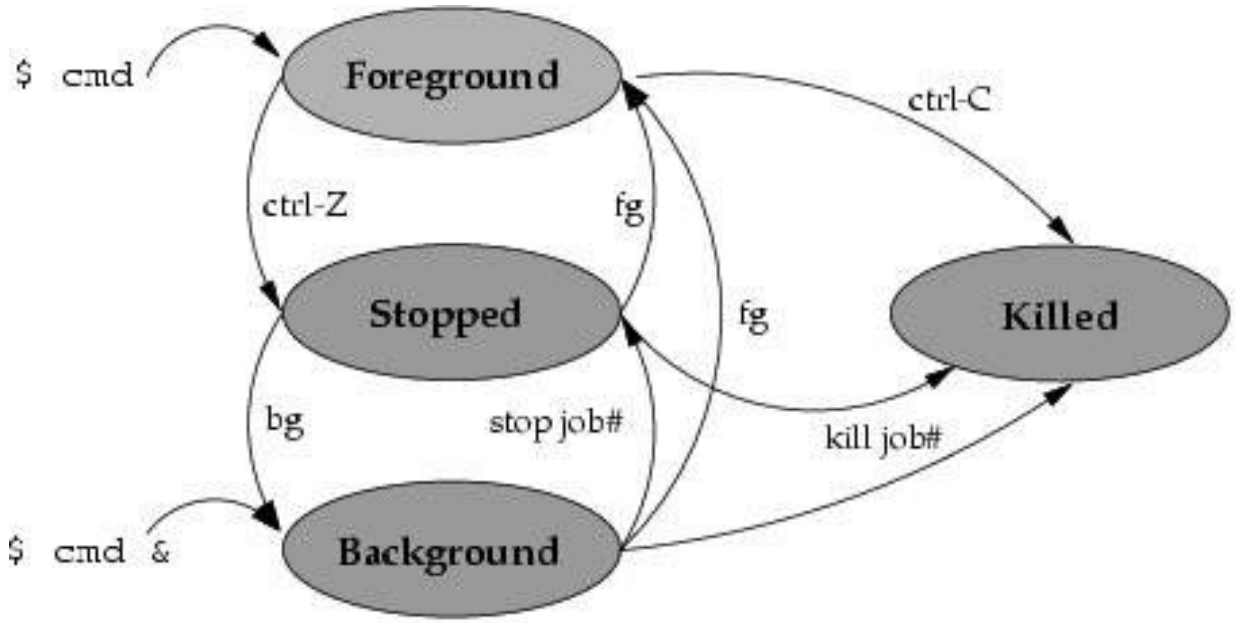
Foreground process can be suspend by ctrl+z and run in background with **bg** or foreground with **fg**.

- **jobs** - display list of jobs.

A job can be chosen by its number in the list with %, %+ for the current job and %- for the previous one:

- **fg** %1 - run job 1 in foreground

Foreground and Background



Foreground and Background processes[2]

[2] https://www.baeldung.com/linux/foreground-background-process

Foreground and Background

Example 3:

- create a new file *notes* using **nano** command and open it in the background.
- Bring it to the foreground, write some notes, save the change, and send it again to the background (Press Ctrl-T + Ctrl-Z).
- Open a new file *notes.txt* using **gedit** in the background. Write something and save the change. Print the status of job list in the shell program.
- Redirect the output of file *notes* to the file *notes.txt*.
- Return to the **gedit** editor and read the notes ("Reload" if it did not show the change).
- Stop both processes.

Exercise 1

Create a directory “week1” in home directory.

- **mkdir ~/week1**
- **cd ~/week1**

List last 5 entries in /usr/bin that contain **“gcc”** in reverse alphabetical order. Save results in

- **“~/week1/ex1.txt”.**

Note: you should submit the file *ex1.txt* and the script *ex1.sh* which contains the commands you’ve run for this exercise.

Exercise 2

Try some commands and save command history to "**~/week1/ex2.txt**". Store the commands in a script *ex2.sh*

Note: you should submit the file *ex2.txt*, and the script *ex2.sh*.

Hints: use *history* command for getting the list of commands executed recently.

Exercise 3

Write a shell script **ex3.sh** that creates two files (*root.txt*, *home.txt*) inside two separate new folders. Before creating the next item (file or folder), print the date and wait for 3 seconds.

The file *root.txt* contains the items of the root directory '/', whereas the file *home.txt* contains the items of the home directory '~'. The items of both directories should be sorted by time (oldest first). Print the content of files and display items of your new folders.

Note: you should submit the files *root.txt* and *home.txt*, and the script **ex3.sh**.

Hints: use the command **date** for getting the current date, and **sleep x** command for pausing the execution x seconds. Run the script with: **sh ex3.sh**.

Exercise 4 - Hello World

Write “Hello world” in the C language. Create source file:
gedit ∼ /*week*1/*main.c*
Write program:

```c
#include <stdio.h>
int main(void)
{
   printf("Hello World!");
}
```

Exercise 4 - Compilation

Compile the program, where ex4 is name of executable file:

- **gcc main.c -o ex4**

Run the program with:

- **./ex4**

Note: you should submit the files **main.c** and **ex4**.

Assignment Submission Instructions

- Create a private repository on Github. One repository is sufficient for the whole course.
- Please make sure that the repo has the following structure:
  - /weekXX/exYY.{c,sh,txt}. e.g. /week06/ex1.c or /week04/ex2.sh.
- Add all TAs to the private repository as collaborators.
  - The TA aliases are in Course Staff & Communication section of this course in Moodle. You can personally ask the TAs about their aliases in case you did not find them.
  - If you did not add a TA to your repo, then the TA can not check your solution and you will get penalized.
- Submit the direct link to the private repository.
  - https://github.com/[username]/[reponame]/tree/master/[weekXX]
- Make sure that your code is readable - that means that the brackets and parenthesis are aligned, the blocks, procedures, loops, etc. have indentations and so on.
- The deadline is the day before your next lecture at 23:55 (the commit time will be checked) so don't hesitate to ask questions.

Useful Links

- About foreground and background processes
- Learning the bash Shell - 3rd Edition
- Design of the Unix Operating System By Maurice Bach
- Console emulator

The End.

Be strong.

Week 01 – Lab 01