Operating Systems Laboratory

Lab 2

Deploying a virtual Linux Workstation

&

Using the fork system call

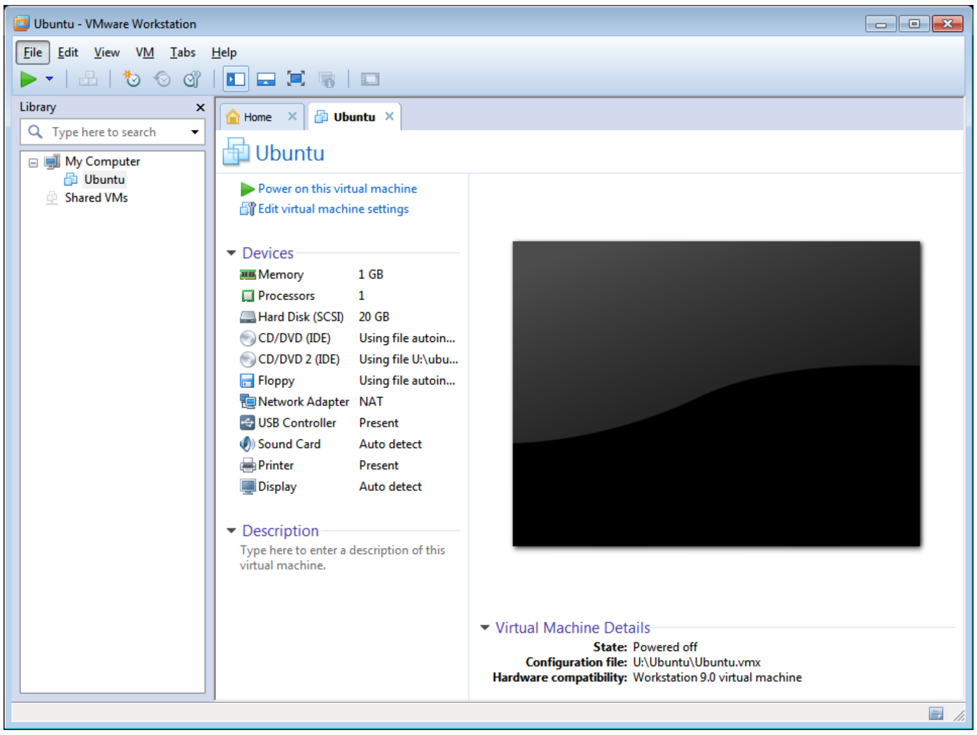
1 Introduction

This lab session is designed to

- Introduce the VMware Workstation and how to deploy a virtual Ubuntu workstation.
- Introduce applications of the fork system call.

2. Deploying a virtual Ubuntu workstation

Double click on the VMware workstation icon and you will see a screen similar to the one in figure 1 appear. Follow File -> New Virtual Workstation.



Ubuntu - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Shared VMs
Home
Ubuntu
Ubuntu
Power on this virtual machine
Edit virtual machine settings
Devices
Description
Type here to enter a description of this virtual machine.
Virtual Machine Details

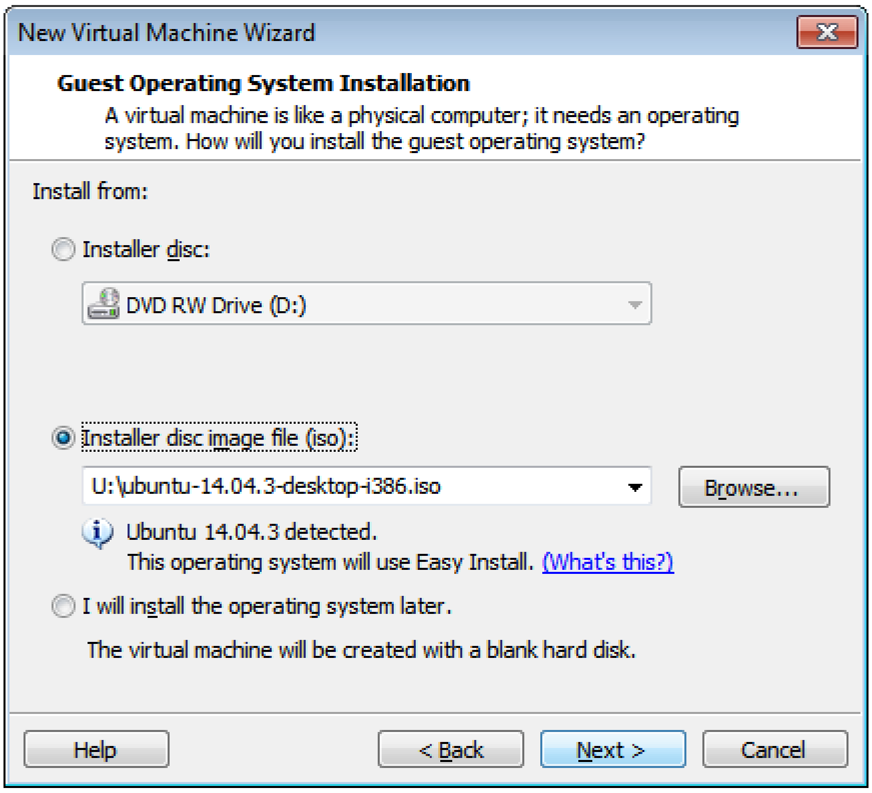
Memory	1 GB
Processors	1
Hard Disk (SCSI)	20 GB
CD/DVD (IDE)	Using file autoin...
CD/DVD 2 (IDE)	Using file U:\ubu...
Floppy	Using file autoin...
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

State: Powered off
Configuration file: U:\Ubuntu\Ubuntu.vmx
Hardware compatibility: Workstation 9.0 virtual machine

Selected “Installer disc image file” and browse to the Ubuntu iso file.

Download the ISO file from http://www.ubuntu.com/download/desktop/thank-you/?version=14.04.3&architecture=i386

Then click next.



New Virtual Machine Wizard

Guest Operating System Installation

A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?

Install from:

○ Installer disc:

DVD RW Drive (D:)

● Installer disc image file (iso):

U:\ubuntu-14.04.3-desktop-i386.iso

Browse...

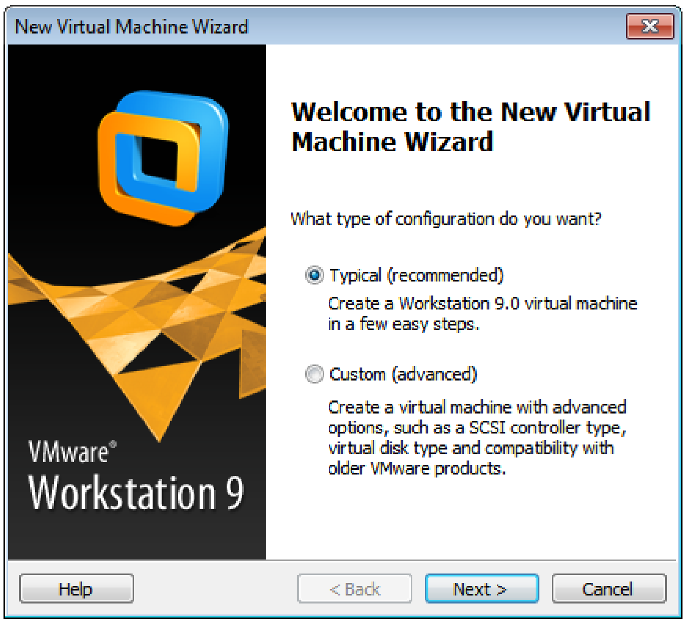
Ubuntu 14.04.3 detected.
This operating system will use Easy Install. (What's this?)

○ I will install the operating system later.

The virtual machine will be created with a blank hard disk.

Help | < Back | Next > | Cancel

When figure 3 appears select typical and then click next.



New Virtual Machine Wizard

VMware® Workstation 9

Welcome to the New Virtual Machine Wizard

What type of configuration do you want?

● Typical (recommended)

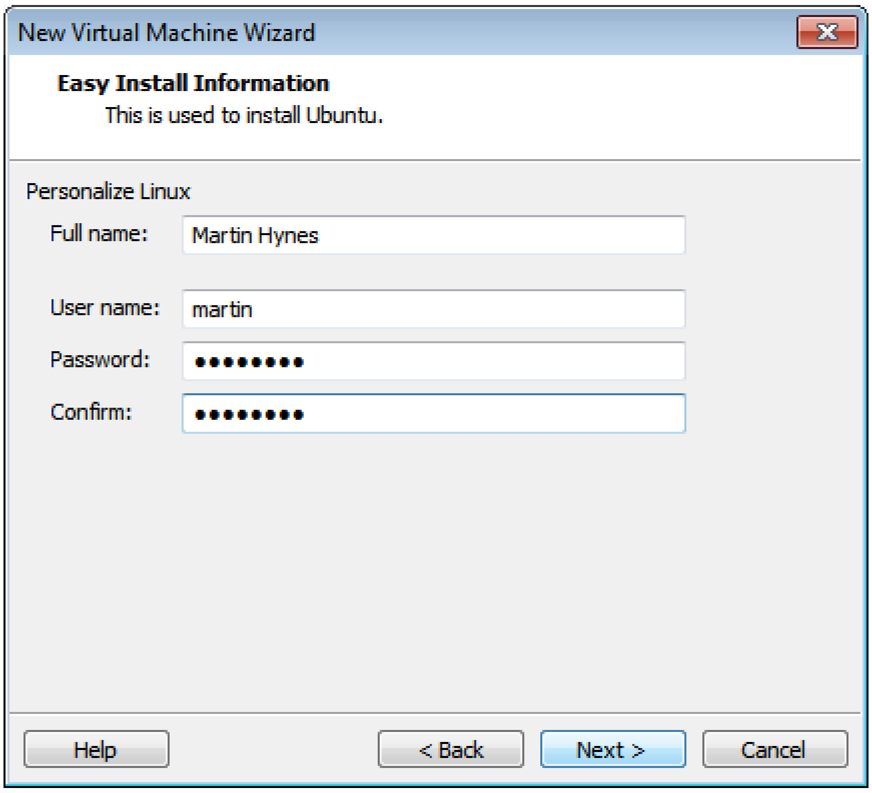
Create a Workstation 9.0 virtual machine in a few easy steps.

○ Custom (advanced)

Create a virtual machine with advanced options, such as a SCSI controller type, virtual disk type and compatibility with older VMware products.

Help | < Back | Next > | Cancel

In the screen shown in Figure 4 you must enter you username and password for the new virtual workstation.


New Virtual Machine Wizard

Easy Install Information
This is used to install Ubuntu.

Personalize Linux

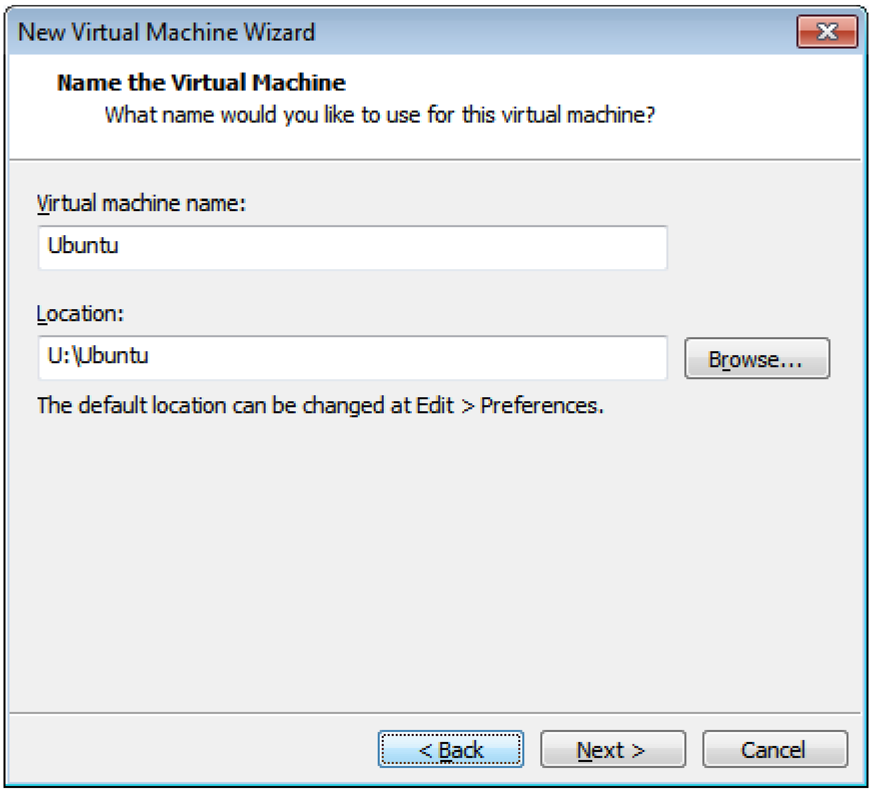
Full name: Martin Hynes

User name: martin

Password: ●●●●●●●●

Confirm: ●●●●●●●●

<u>Then you must select the directory of the virtual machine. Ensure you select a folder in the U drive</u>


New Virtual Machine Wizard

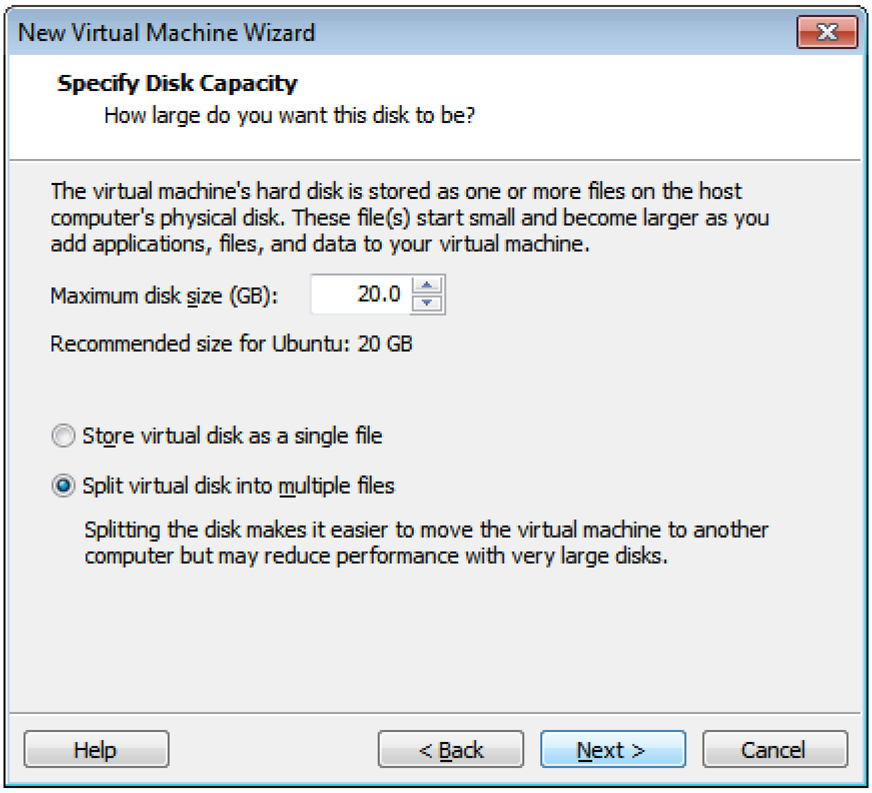
Name the Virtual Machine
What name would you like to use for this virtual machine?

Virtual machine name: Ubuntu

Location: U:\Ubuntu

The default location can be changed at Edit > Preferences.

Then click next and finish. You must then wait to create the new virtual machine.



New Virtual Machine Wizard

Specify Disk Capacity

How large do you want this disk to be?

The virtual machine's hard disk is stored as one or more files on the host computer's physical disk. These file(s) start small and become larger as you add applications, files, and data to your virtual machine.

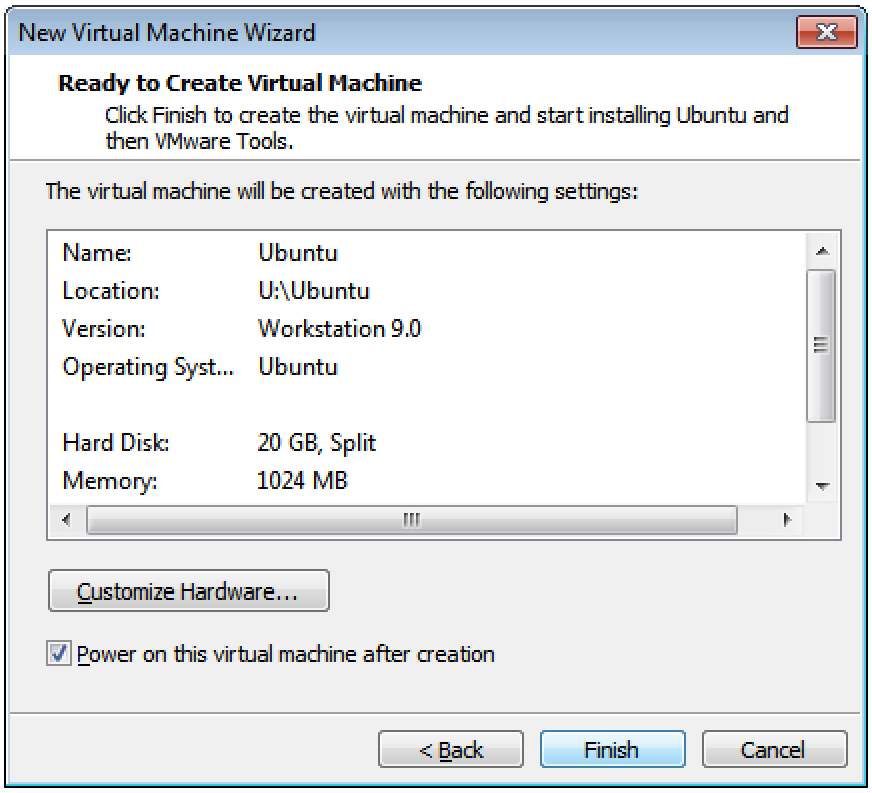
Maximum disk size (GB): 20.0

Recommended size for Ubuntu: 20 GB

○ Store virtual disk as a single file

● Split virtual disk into multiple files

Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.



New Virtual Machine Wizard

Ready to Create Virtual Machine

Click Finish to create the virtual machine and start installing Ubuntu and then VMware Tools.

The virtual machine will be created with the following settings:

Name:	Ubuntu
Location:	U:\Ubuntu
Version:	Workstation 9.0
Operating Syst...	Ubuntu
Hard Disk:	20 GB, Split
Memory:	1024 MB

☑ Power on this virtual machine after creation

2. Working with the Fork System Call

Most operating systems identify processes according to a unique process identifier (pid), which is typically an integer number.

In Unix the system call fork is used to create a new process, which becomes the child process of the caller. Both processes will execute concurrently from the next instruction following the fork() system call. We have to distinguish the parent from the child. This can be done by testing the returned value of fork() such it returns pid with value:

1. -1 if an error happened and fork() failed.
2. 0 to the child process.
3. Positive Process id of child to the parent.

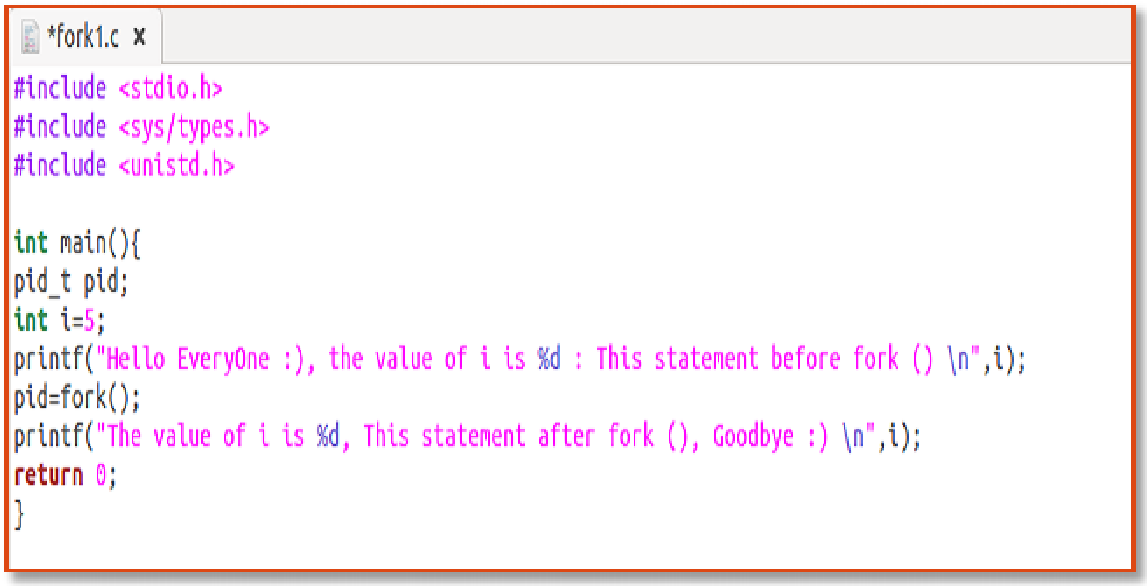
When a process creates a new process, two possibilities exist in terms of execution:

1. The parent continues to execute concurrently with its children.
2. The parent waits until some or all of its children have terminated.

There are also two possibilities in terms of the address space of the new process:

1. The child process is a duplicate of the parent process (it has the same program and data as the parent).
2. The child process has a new program loaded into it.

Example 1



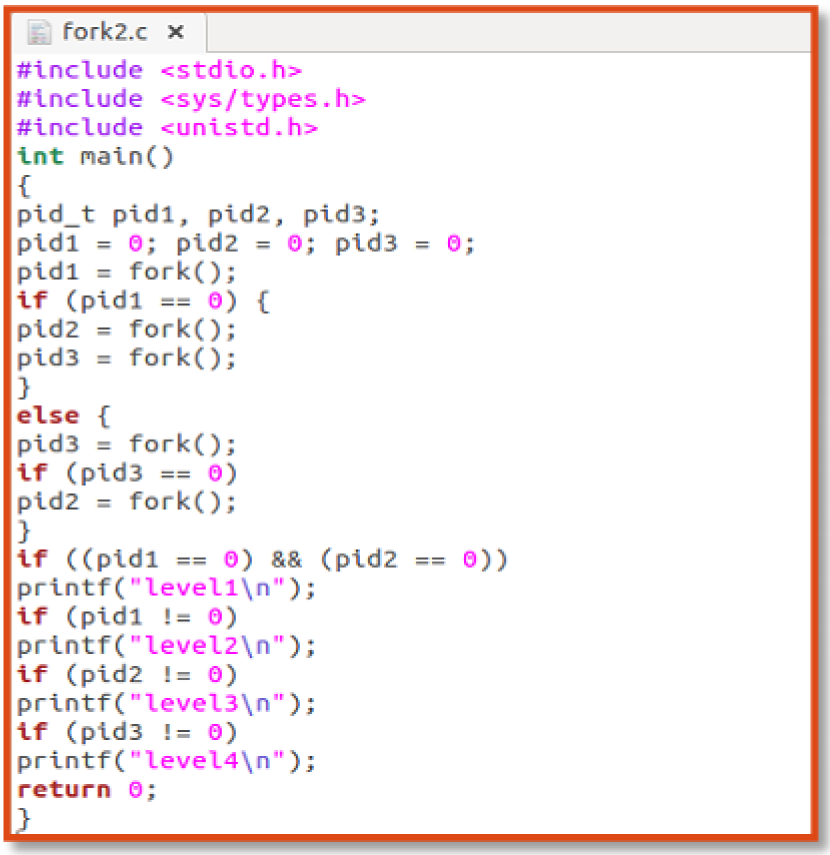
*fork1.c

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main(){
pid_t pid;
int i=5;
printf("Hello EveryOne :), the value of i is %d : This statement before fork () \n",i);
pid=fork();
printf("The value of i is %d, This statement after fork (), Goodbye :) \n",i);
return 0;
}
```

Practical:

Please explain the output of the program?

Example 2

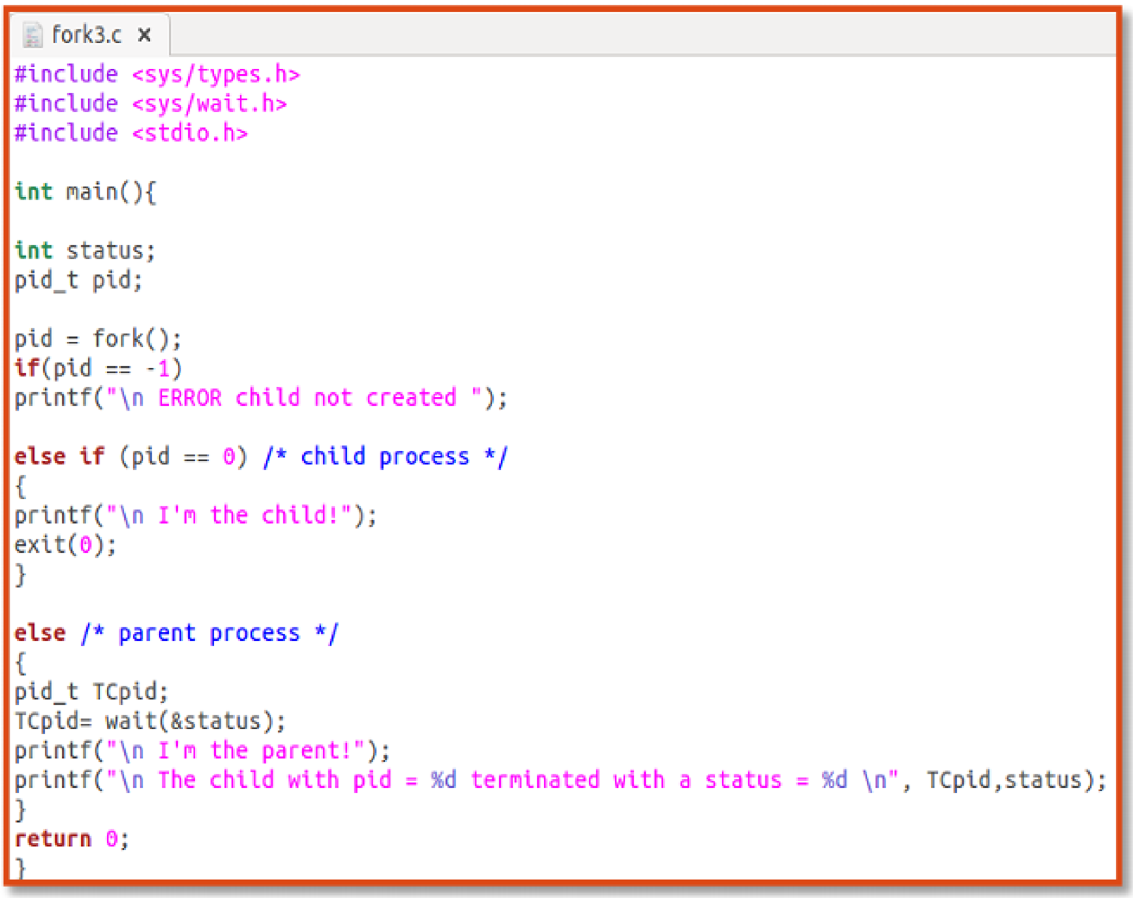


```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
pid_t pid1, pid2, pid3;
pid1 = 0; pid2 = 0; pid3 = 0;
pid1 = fork();
if (pid1 == 0) {
pid2 = fork();
pid3 = fork();
}
else {
pid3 = fork();
if (pid3 == 0)
pid2 = fork();
}
if ((pid1 == 0) && (pid2 == 0))
printf("level1\n");
if (pid1 != 0)
printf("level2\n");
if (pid2 != 0)
printf("level3\n");
if (pid3 != 0)
printf("level4\n");
return 0;
}
```

Practical:

Please explain the output of the program?

Example 3

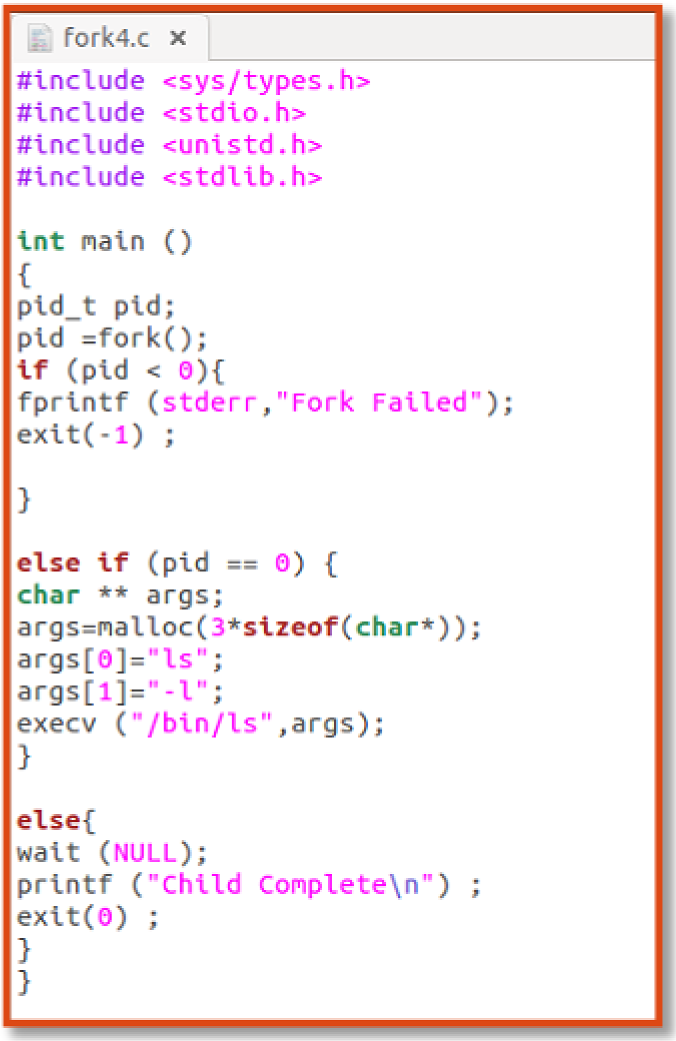


```
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>

int main(){

int status;
pid_t pid;

pid = fork();
if(pid == -1)
printf("\n ERROR child not created ");

else if (pid == 0) /* child process */
{
printf("\n I'm the child!");
exit(0);
}

else /* parent process */
{
pid_t TCpid;
TCpid= wait(&status);
printf("\n I'm the parent!");
printf("\n The child with pid = %d terminated with a status = %d \n", TCpid,status);
}
return 0;
}
```

<u>Practical:</u>

<u>Can you explain what happened in example 3?</u>

Example 4



```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main ()
{
pid_t pid;
pid =fork();
if (pid < 0){
fprintf (stderr,"Fork Failed");
exit(-1) ;

}

else if (pid == 0) {
char ** args;
args=malloc(3*sizeof(char*));
args[0]="ls";
args[1]="-l";
execv ("/bin/ls",args);
}

else{
wait (NULL);
printf ("Child Complete\n") ;
exit(0) ;
}
}
```

<u>Practical:</u>

<u>Can you explain what happened in example 4?</u>