# Lab 1: Introduction to CLI/Batch in Windows

**Introduction:**

Majority of the students with already computer background may be familiar with this, but just align all the students, this is a first exercise to practice the basics of DOS Scripting.

**Objective:**

To familiarize yourself with Windows Command Line and Batch scripting

**Tools/Software Requirement:**

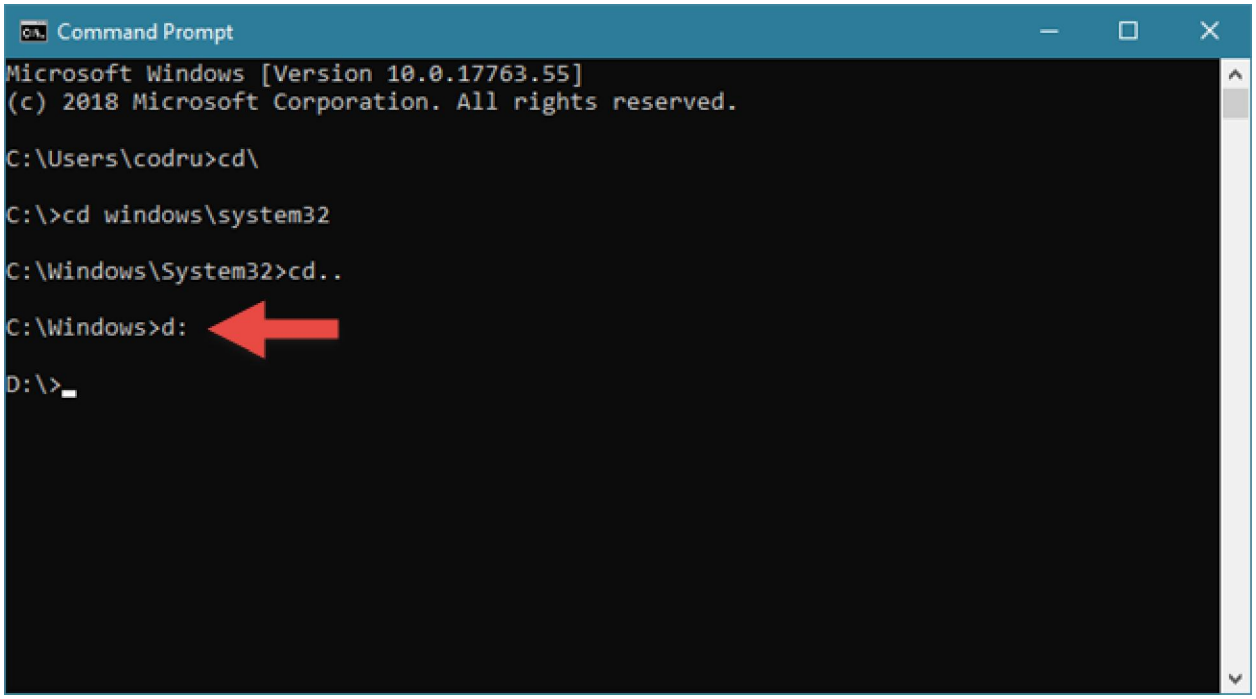
- Windows / Command Prompt

**Description/Tasks:**

a. Write DOS basic commands, such as copy/paste, rename, directory navigation.
b. Write a batch file to develop a backup utility.

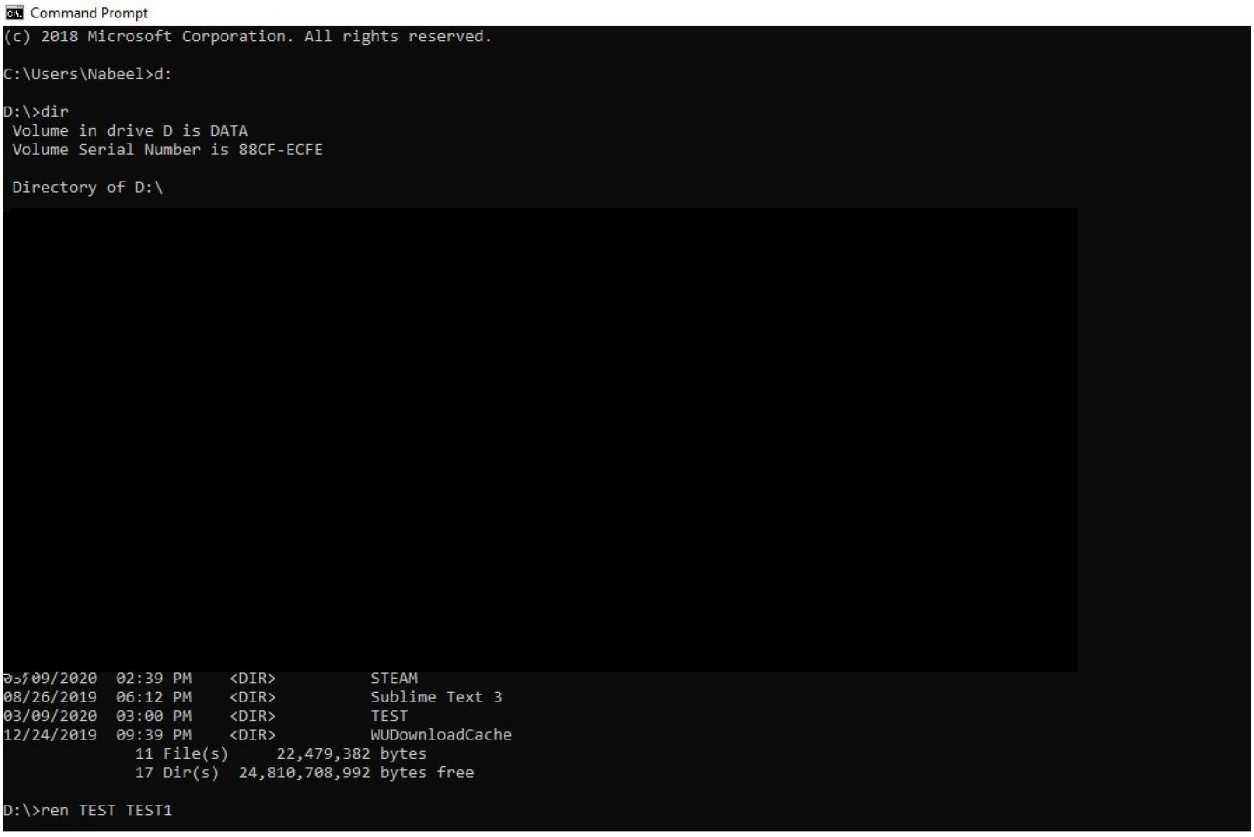
**Implementation:**

- Directory navigation.




- Copy/Paste

- Rename




## Lab 2: Introduction to Ubuntu

**Introduction:**
Introduce and motivate you regarding Open-source operating systems, its usage in Server as well as desktop computing.

**Objective:**
To know about the programs available in Ubuntu and its comparison with Windows programs

**Tools/Software Requirement:**

- Ubuntu Operating system

**Description/Tasks:**

- Basic Programs
    - Office Suite

- Web Browser:
- Databases

- Directory Structure
  - How Ubuntu is different from windows directory structure

- Installation of Ubuntu

  For all subsequent labs, you will be requiring Ubuntu, so learn how to install

  a. Vanish windows from your computer and install Ubuntu from scratch.

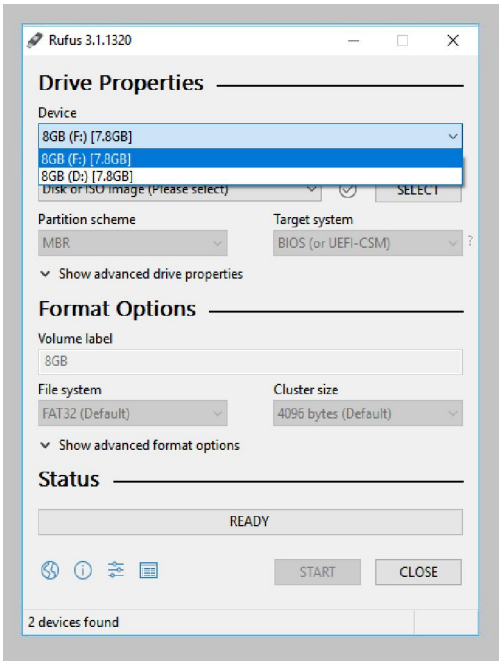
  b. Dual boot Ubuntu and Windows

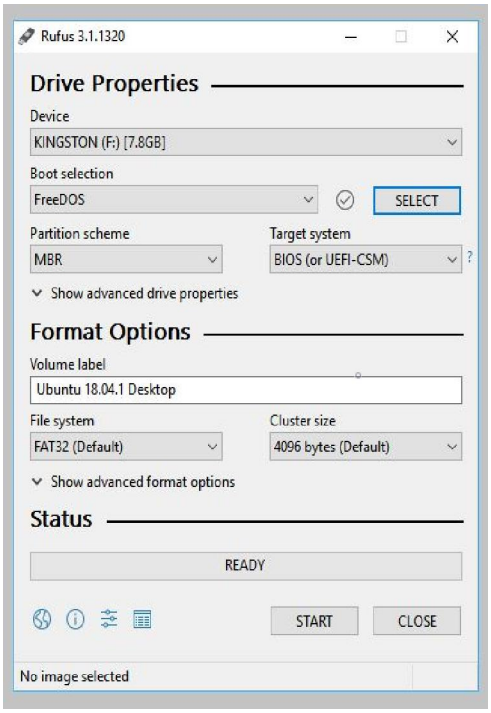
  c. Run Ubuntu inside a virtual environment such as Oracle Virtual Box, VMWare

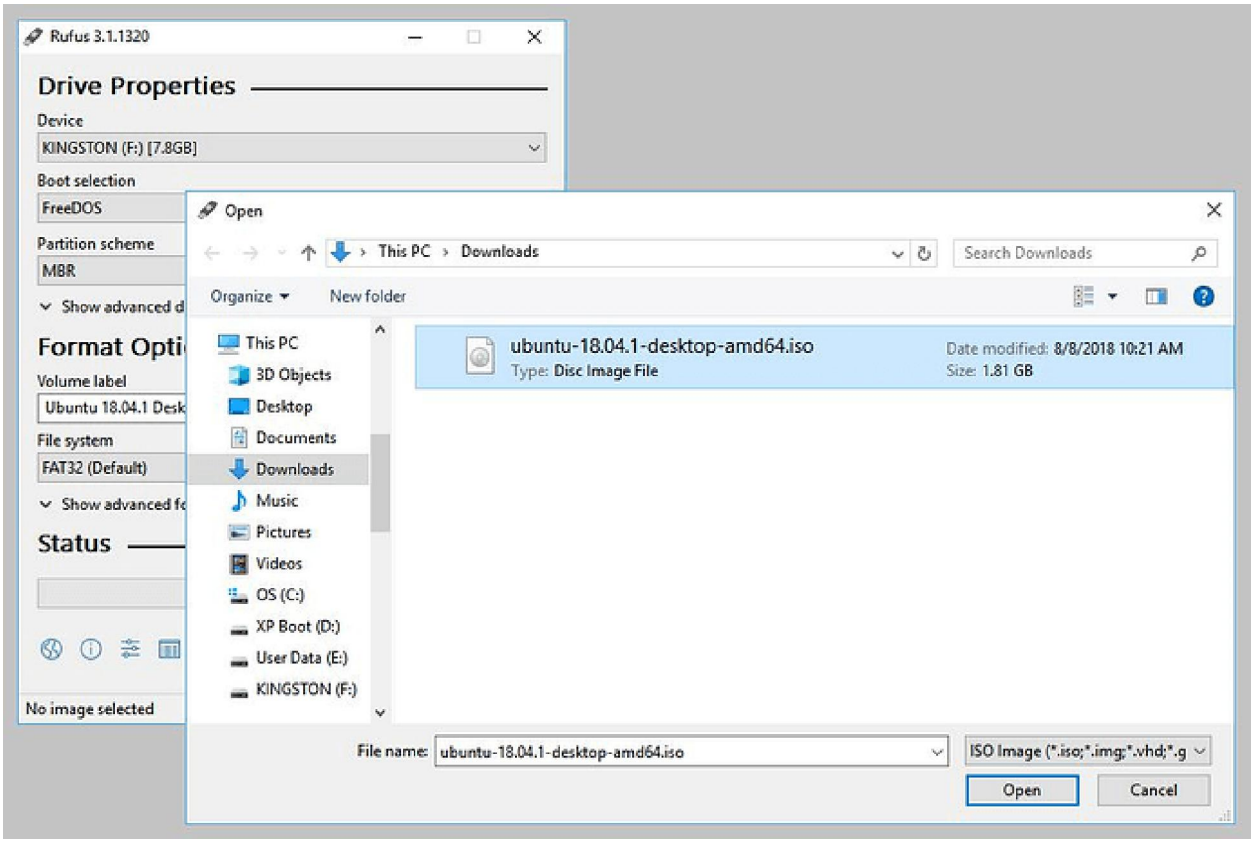
**Implementation:**

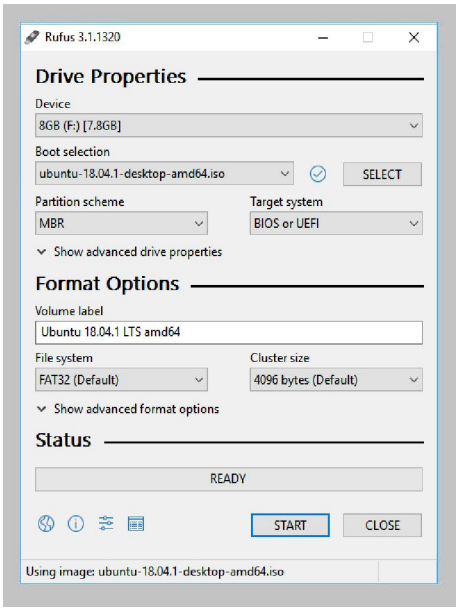
- Dual boot Ubuntu and Windows.

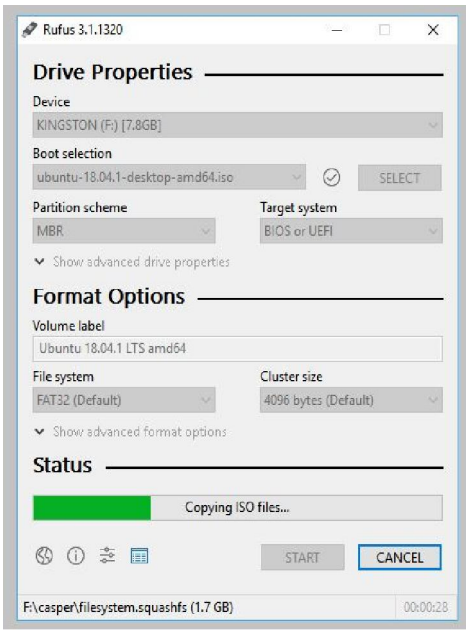
Download the Ubuntu ISO file from: https://ubuntu.com/#download
Get RUFUS from: https://rufus.ie/



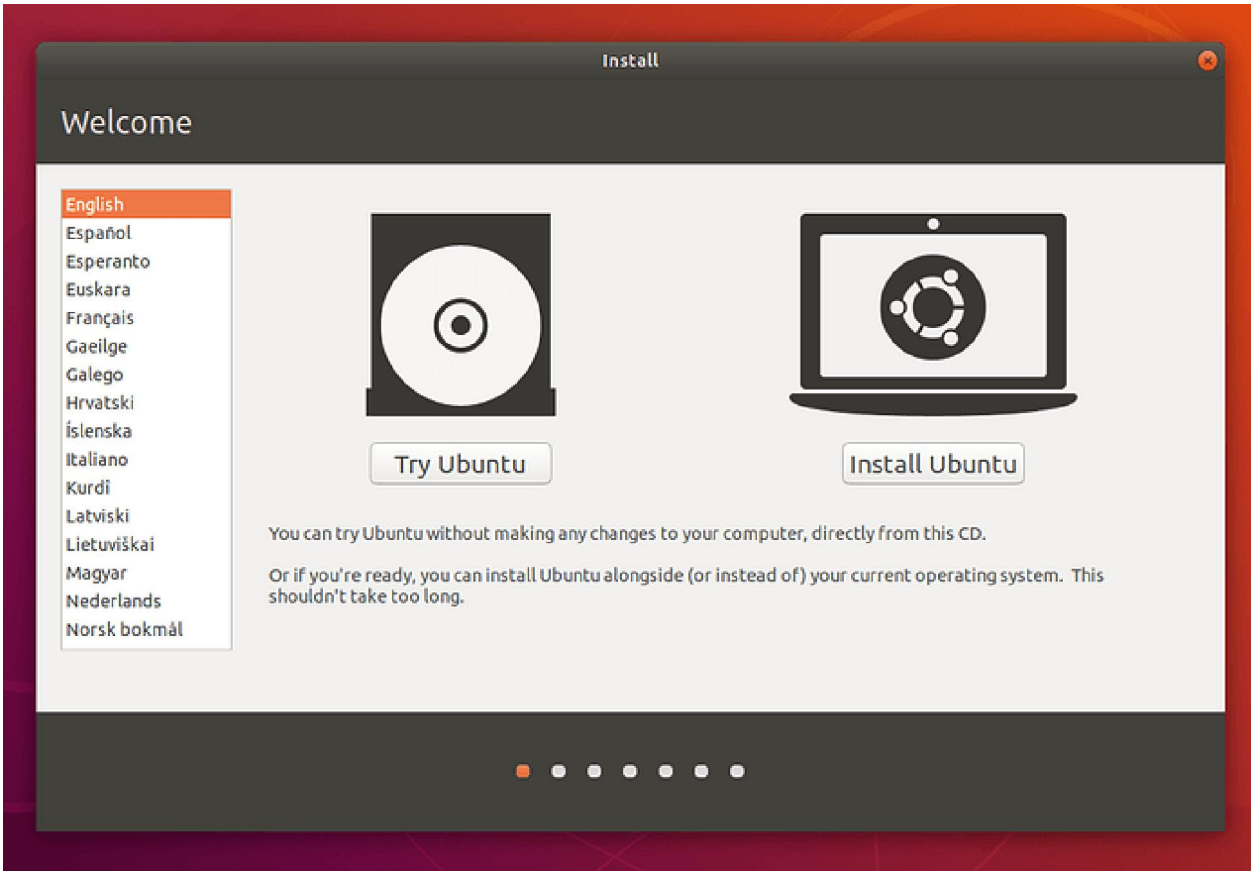

Rufus 3.1.1320
Drive Properties
Device
KINGSTON (F:) [7.8GB]
Boot selection
FreeDOS
SELECT
Partition scheme
MBR
Target system
BIOS (or UEFI-CSM)
Show advanced drive properties
Format Options
Volume label
Ubuntu 18.04.1 Desktop
File system
FAT32 (Default)
Cluster size
4096 bytes (Default)
Show advanced format options
Status
READY
START
CLOSE
No image selected

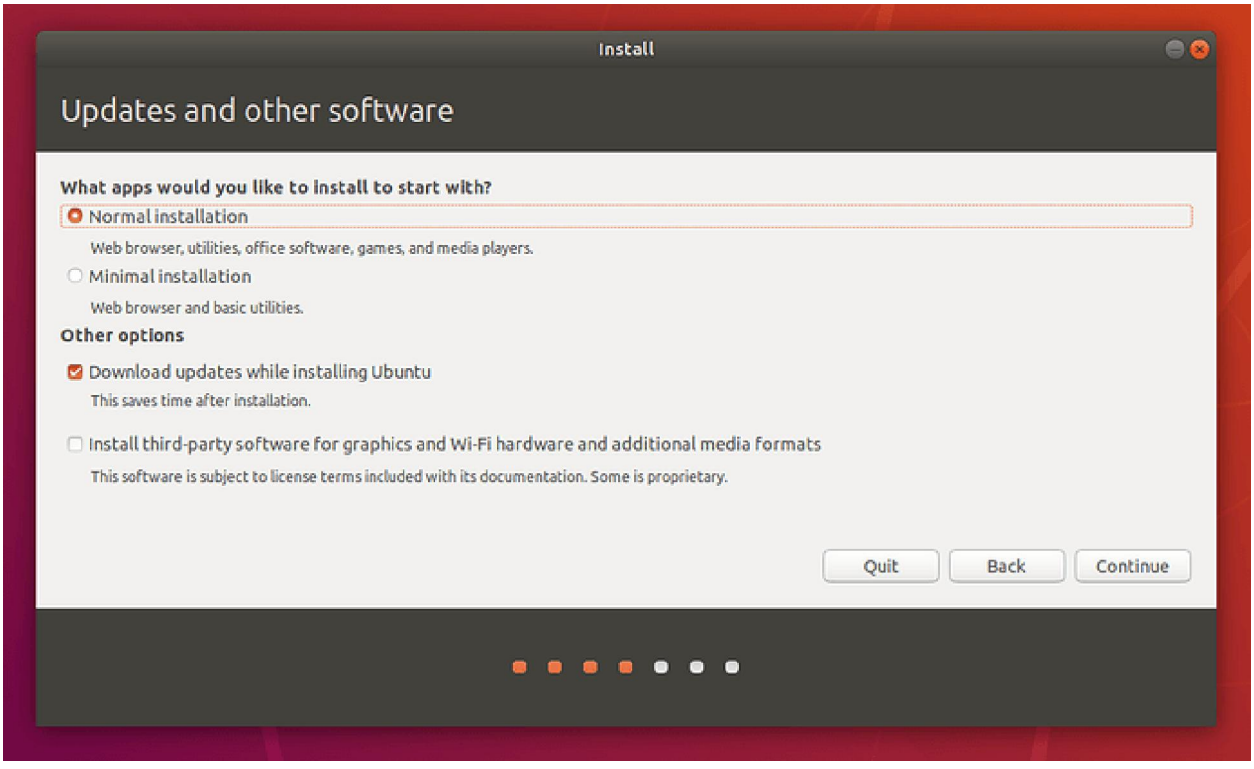

Rufus 3.1.1320
Drive Properties
Device
KINGSTON (F:) [7.8GB]
Boot selection
FreeDOS
Partition scheme
MBR
Format Opti
Volume label
Ubuntu 18.04.1 Desk
File system
FAT32 (Default)
Status
No image selected
Open
This PC
Downloads
Search Downloads
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
XP Boot (D:)
User Data (E:)
KINGSTON (F:)
ubuntu-18.04.1-desktop-amd64.iso
Type: Disc Image File
Date modified: 8/8/2018 10:21 AM
Size: 1.81 GB
File name:
ubuntu-18.04.1-desktop-amd64.iso
ISO Image (*.iso;*.img;*.vhd;*.g
Open
Cancel

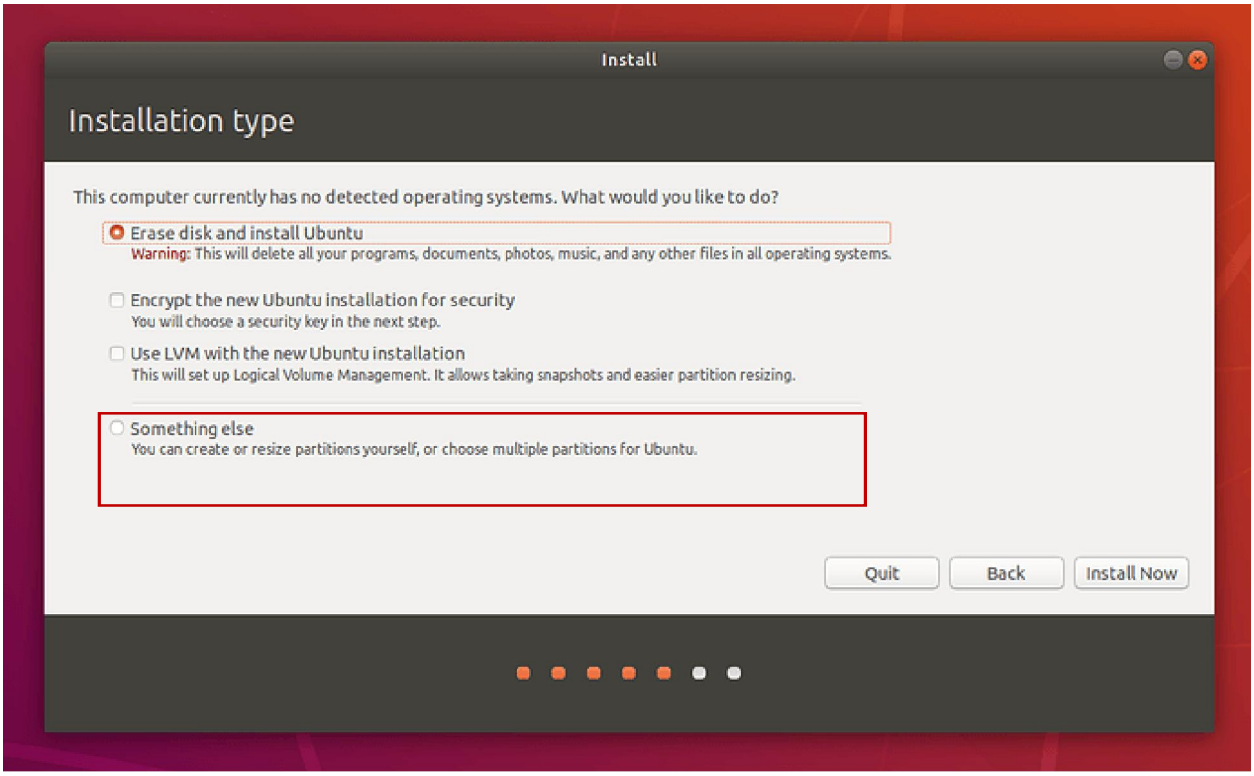


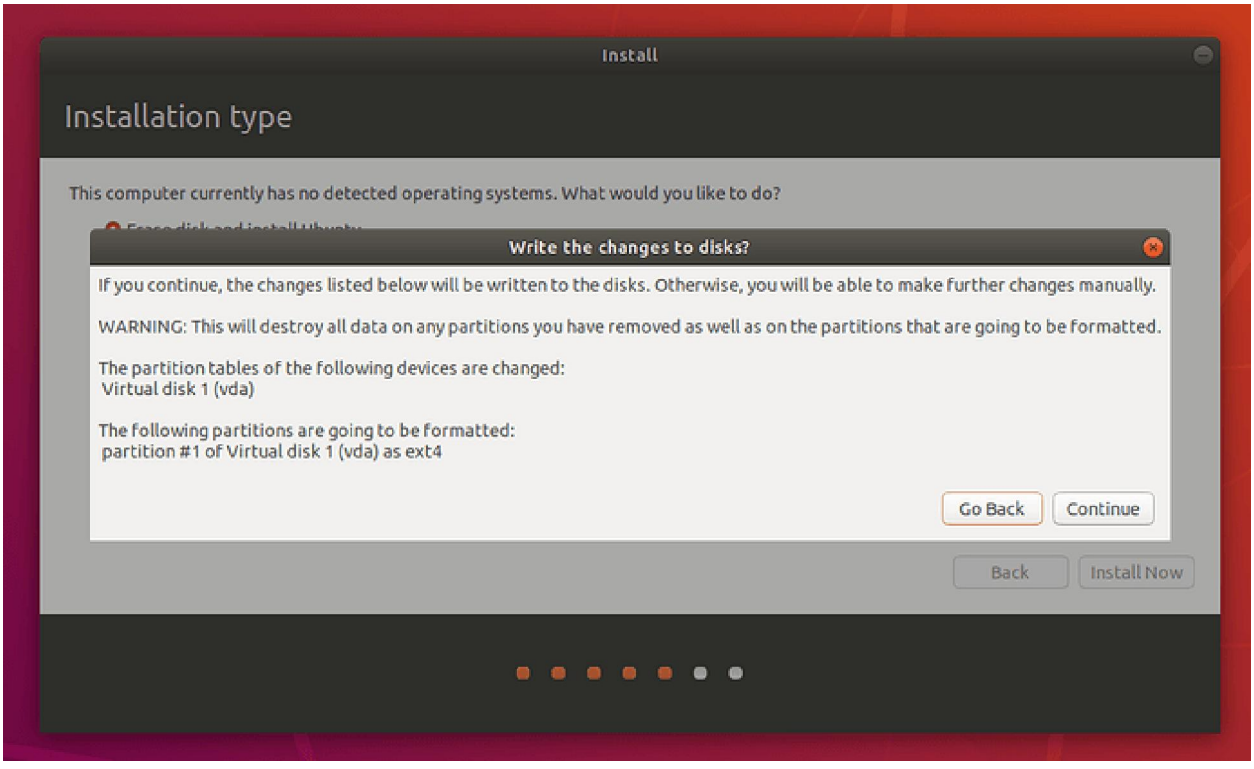



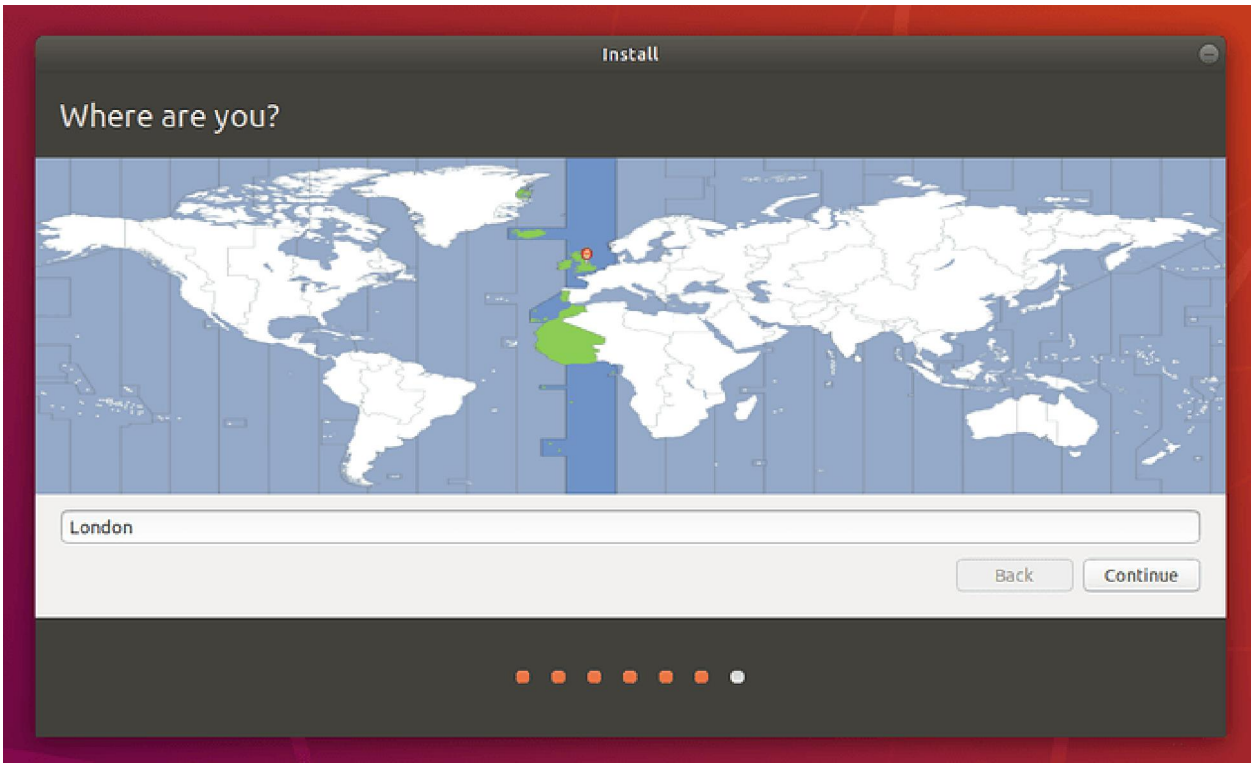

Boot from USB.

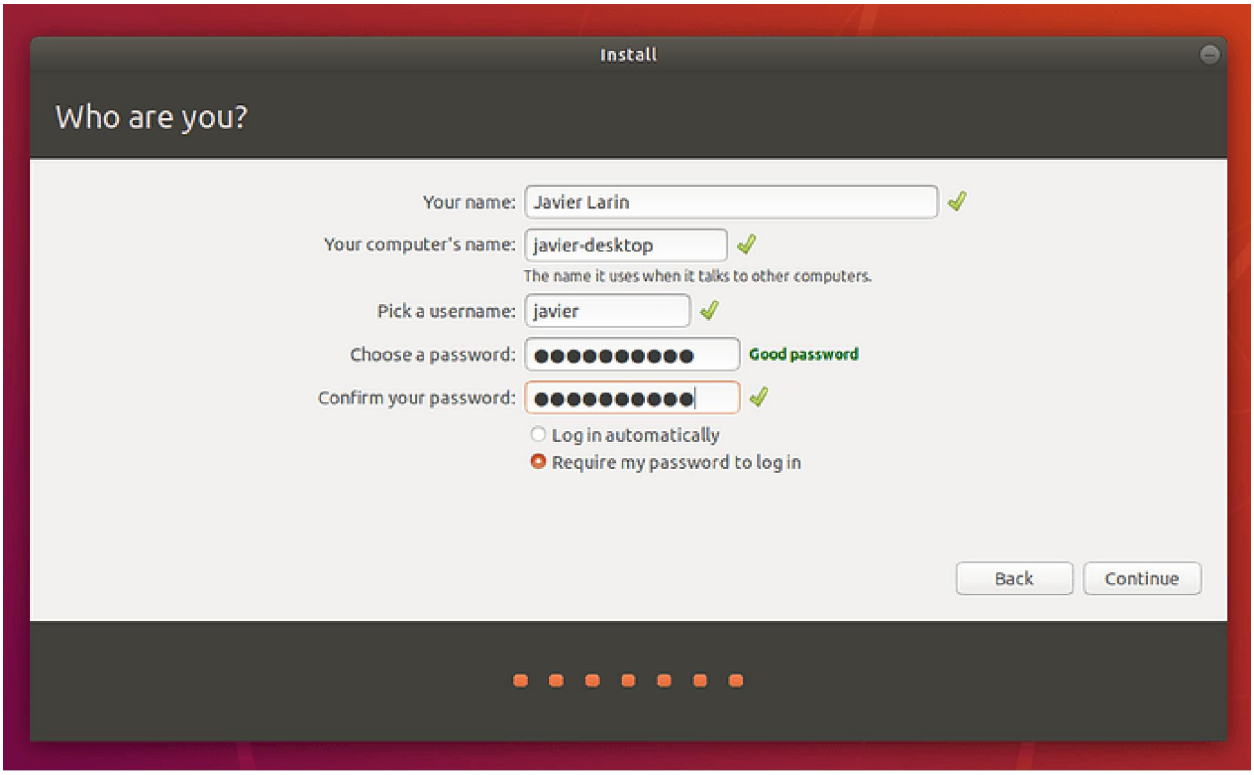

Install
Welcome
English
Español
Esperanto
Euskara
Français
Gaeilge
Galego
Hrvatski
Íslenska
Italiano
Kurdî
Latviski
Lietuviškai
Magyar
Nederlands
Norsk bokmål
Try Ubuntu
Install Ubuntu
You can try Ubuntu without making any changes to your computer, directly from this CD.
Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.

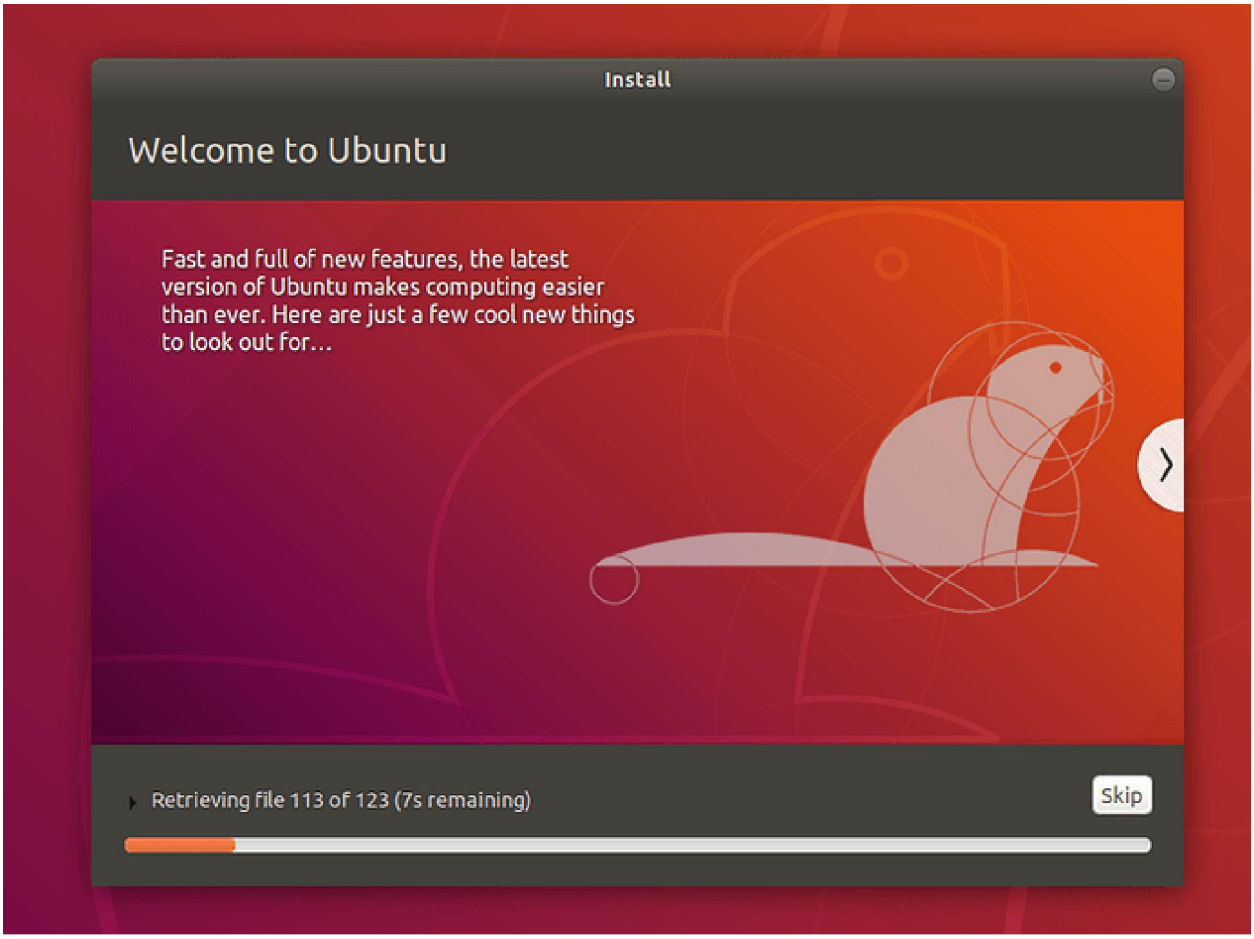

Install
Updates and other software
What apps would you like to install to start with?
Normal installation
Web browser, utilities, office software, games, and media players.
Minimal installation
Web browser and basic utilities.
Other options
Download updates while installing Ubuntu
This saves time after installation.
Install third-party software for graphics and Wi-Fi hardware and additional media formats
This software is subject to license terms included with its documentation. Some is proprietary.
Quit
Back
Continue


Install
Installation type
This computer currently has no detected operating systems. What would you like to do?
Erase disk and install Ubuntu
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.
Encrypt the new Ubuntu installation for security
You will choose a security key in the next step.
Use LVM with the new Ubuntu installation
This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.
Something else
You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.
Quit
Back
Install Now

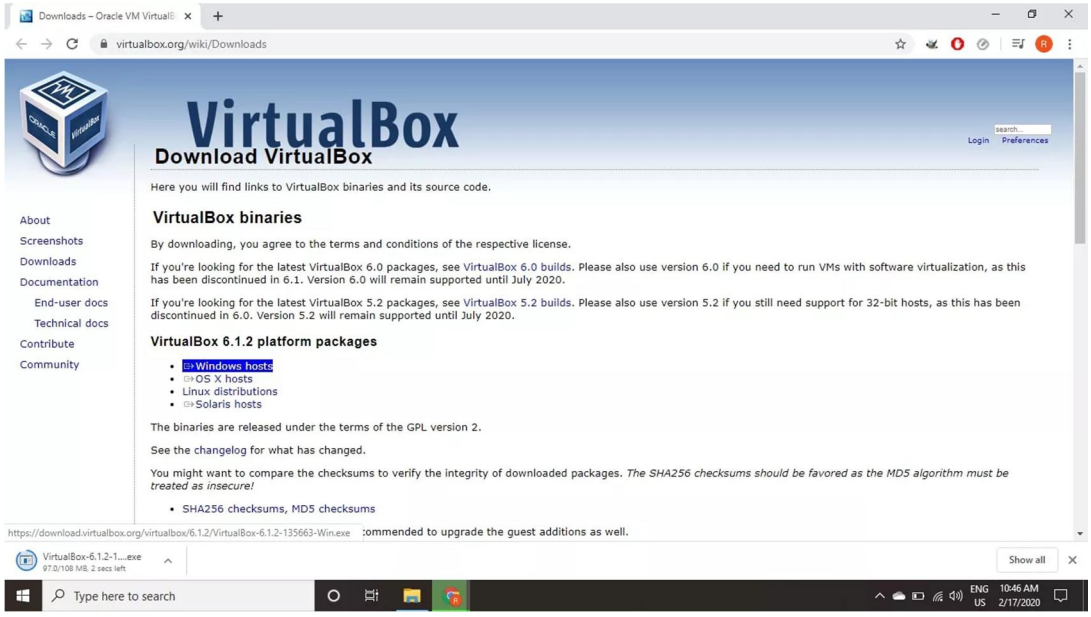

Install
Installation type
This computer currently has no detected operating systems. What would you like to do?
Write the changes to disks?
If you continue, the changes listed below will be written to the disks. Otherwise, you will be able to make further changes manually.
WARNING: This will destroy all data on any partitions you have removed as well as on the partitions that are going to be formatted.
The partition tables of the following devices are changed:
Virtual disk 1 (vda)
The following partitions are going to be formatted:
partition #1 of Virtual disk 1 (vda) as ext4
Go Back
Continue
Back
Install Now

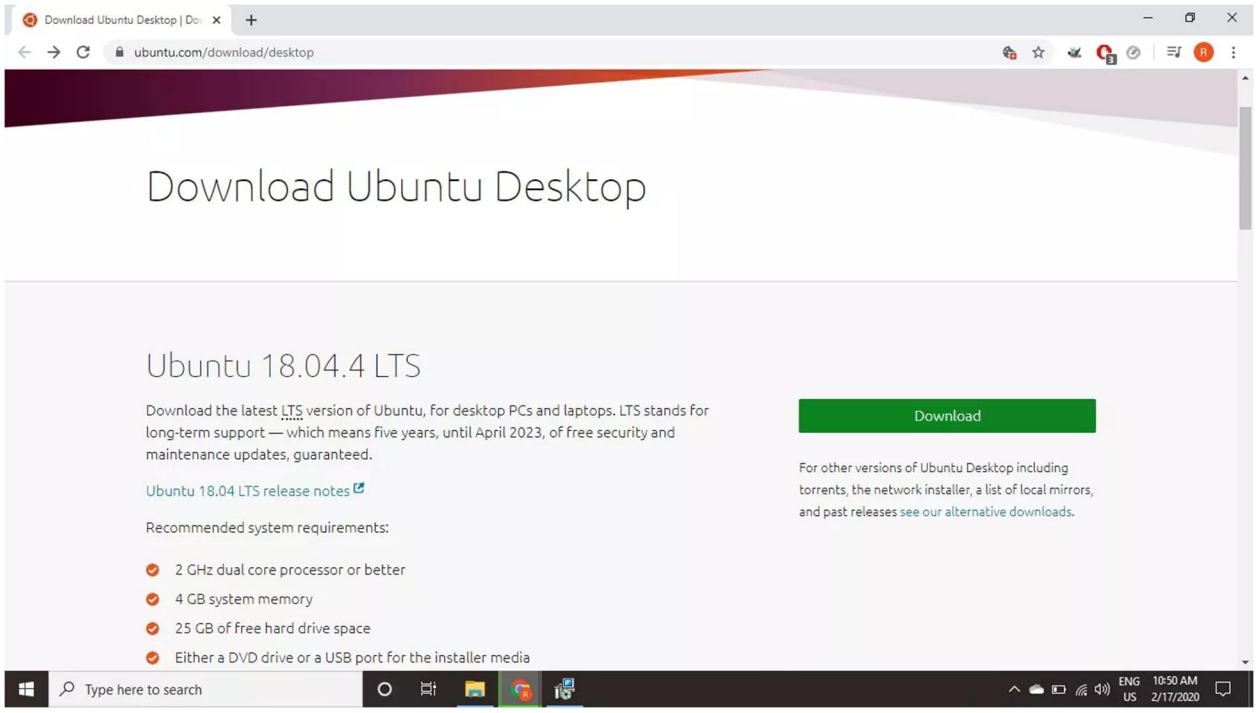

Install
Where are you?
London
Back
Continue

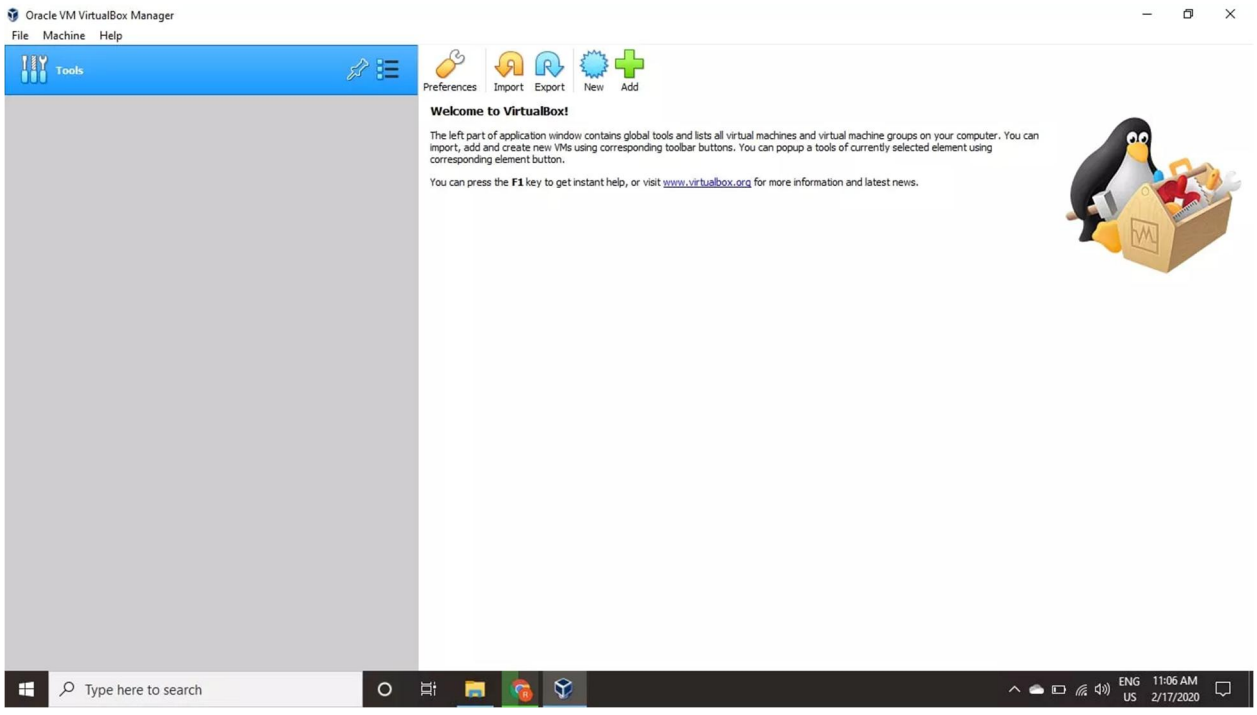

Install
Who are you?
Your name:
Javier Larin
Your computer's name:
javier-desktop
The name it uses when it talks to other computers.
Pick a username:
javier
Choose a password:
Good password
Confirm your password:
Log in automatically
Require my password to log in
Back
Continue

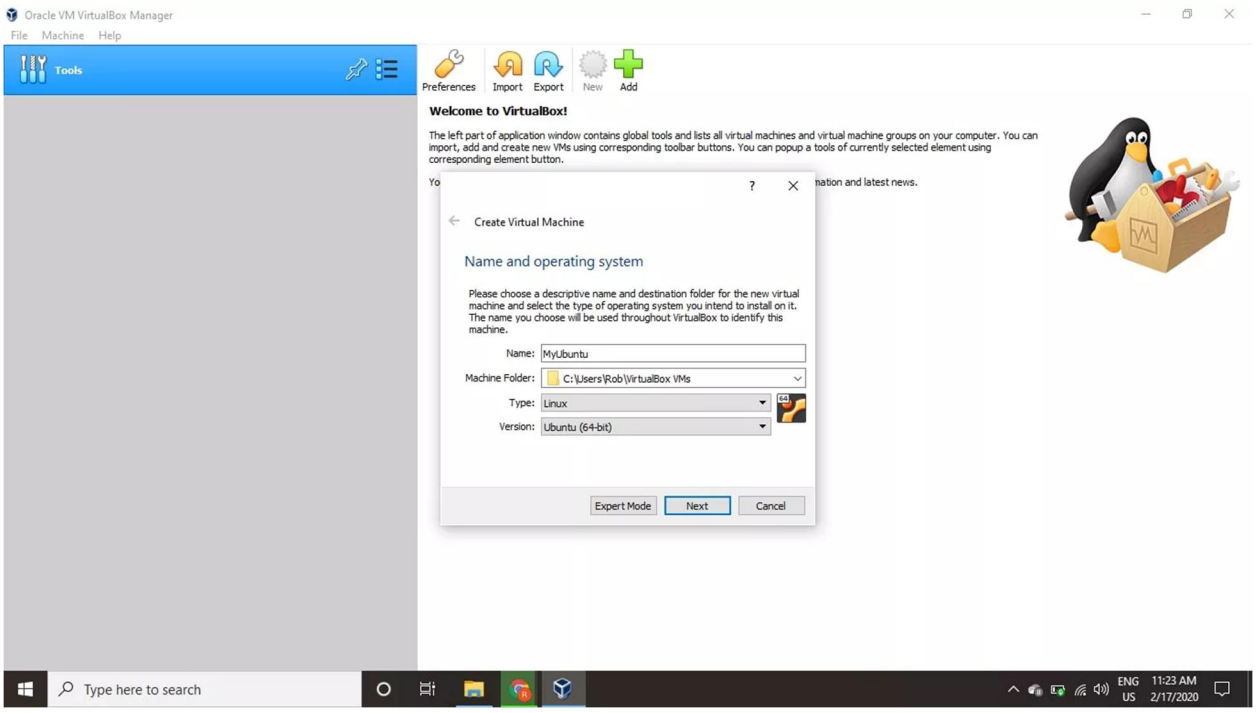


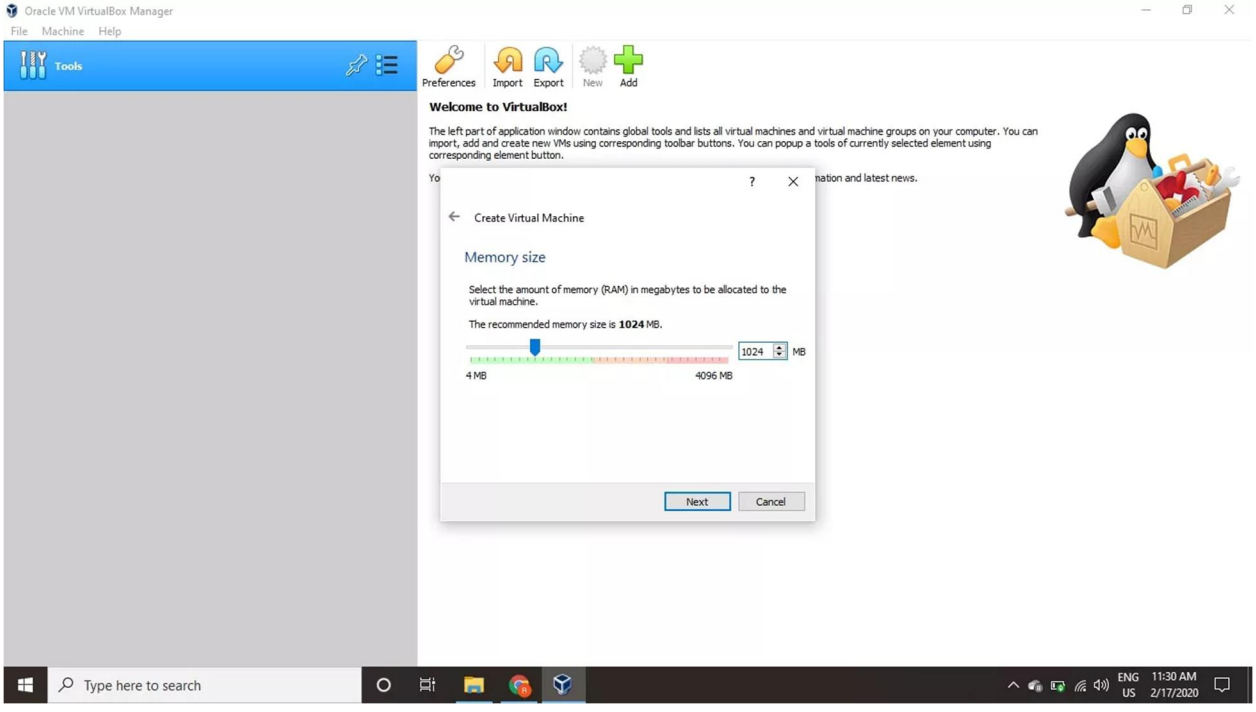

- Run Ubuntu inside a virtual environment.

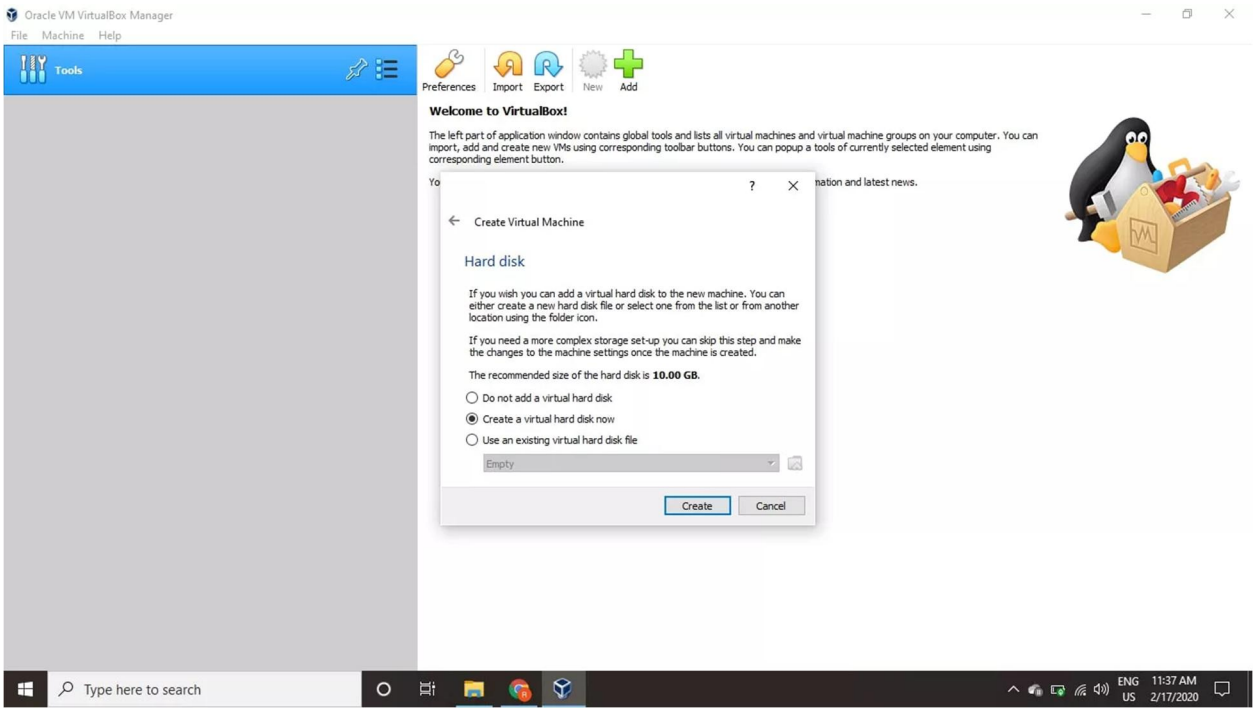


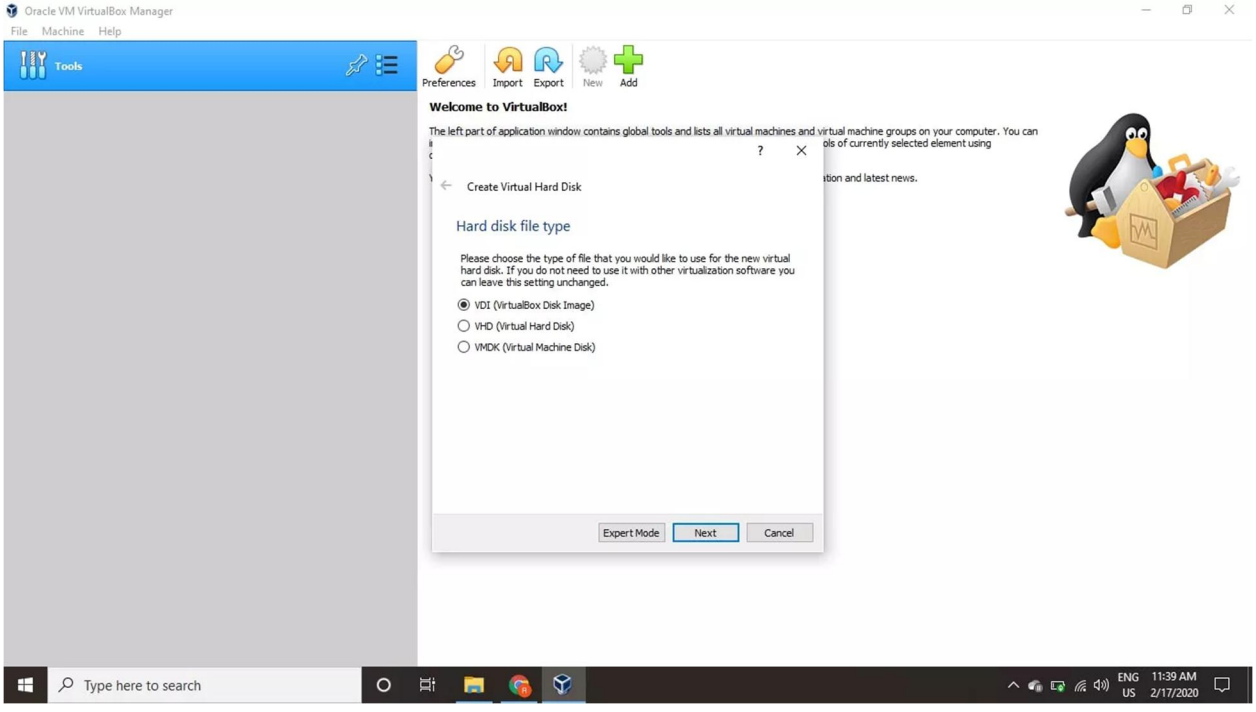

Download Ubuntu Desktop | Do
ubuntu.com/download/desktop
Download Ubuntu Desktop
Ubuntu 18.04.4 LTS
Download the latest LTS version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support — which means five years, until April 2023, of free security and maintenance updates, guaranteed.
Ubuntu 18.04 LTS release notes
Recommended system requirements:
2 GHz dual core processor or better
4 GB system memory
25 GB of free hard drive space
Either a DVD drive or a USB port for the installer media
Download
For other versions of Ubuntu Desktop including torrents, the network installer, a list of local mirrors, and past releases see our alternative downloads.
Type here to search
ENG US
10:50 AM
2/17/2020

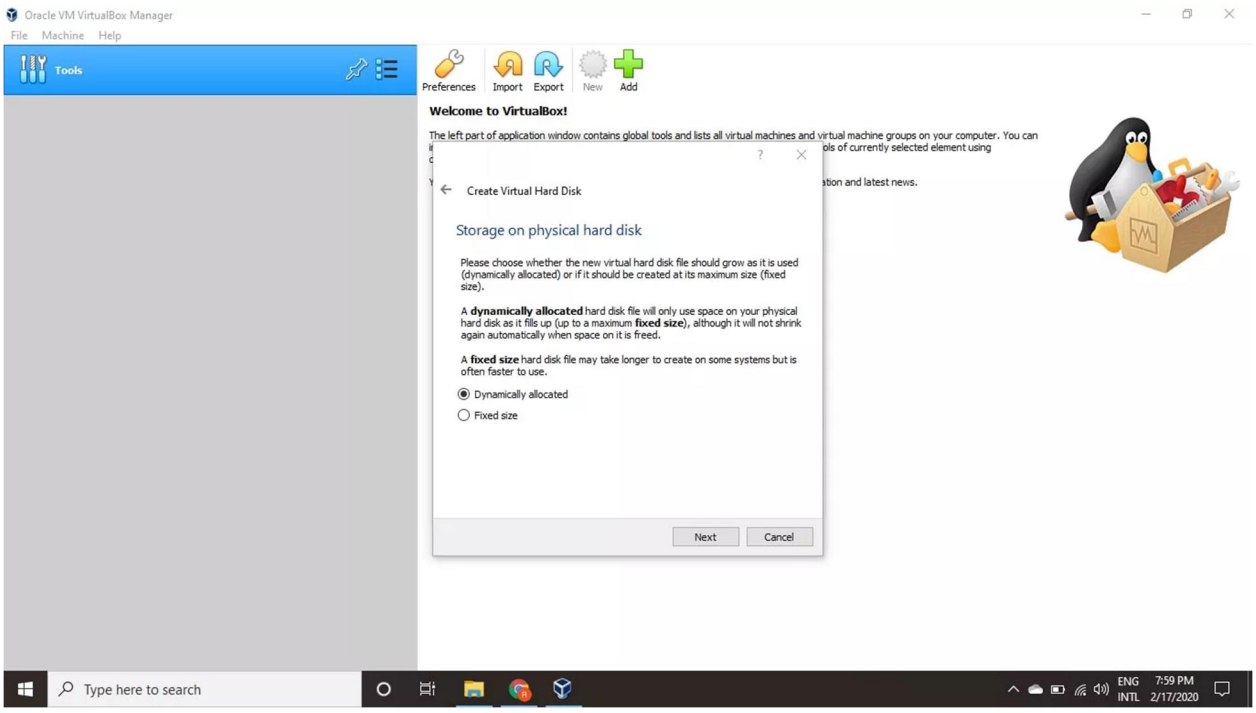

Oracle VM VirtualBox Manager
File Machine Help
Tools
Preferences
Import
Export
New
Add
Welcome to VirtualBox!
The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.
You can press the F1 key to get instant help, or visit www.virtualbox.org for more information and latest news.
Type here to search
ENG US
11:06 AM
2/17/2020

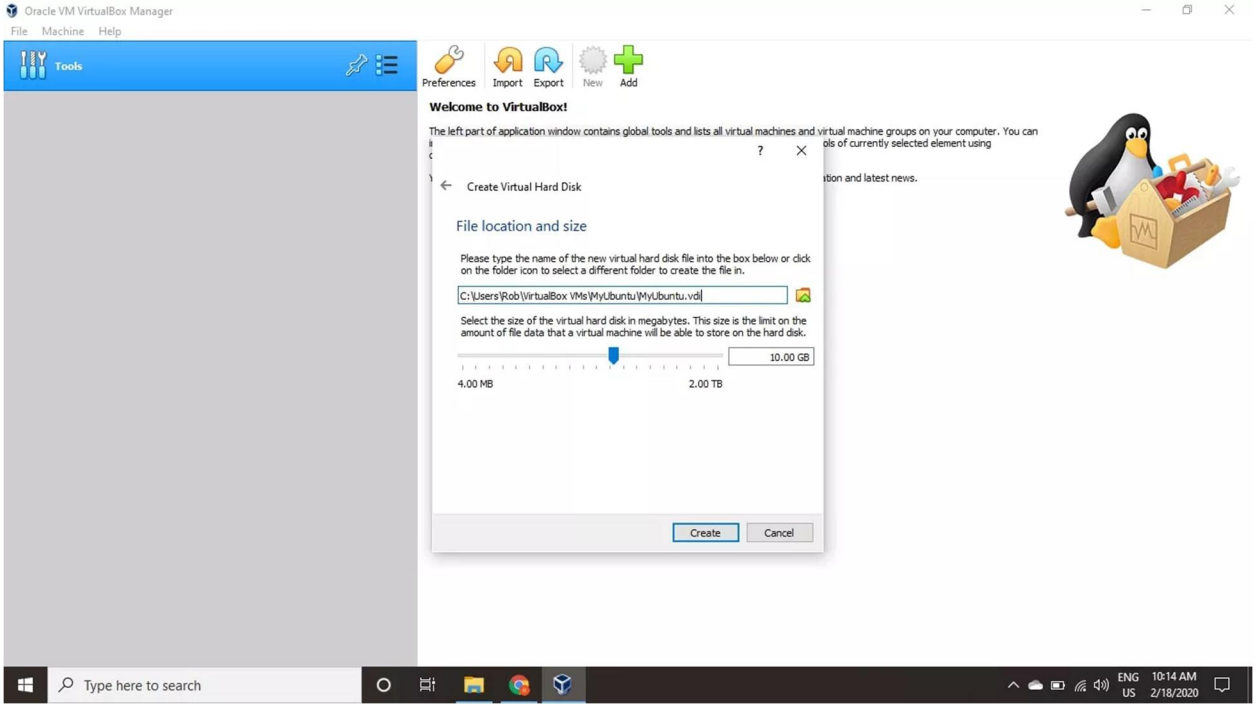

Oracle VM VirtualBox Manager
File Machine Help
Tools
Preferences Import Export New Add
Welcome to VirtualBox!
The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.
nation and latest news.
Create Virtual Machine
Name and operating system
Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.
Name: MyUbuntu
Machine Folder: C:\Users\Rob\VirtualBox VMs
Type: Linux
Version: Ubuntu (64-bit)
Expert Mode Next Cancel
Type here to search
ENG US 11:23 AM 2/17/2020

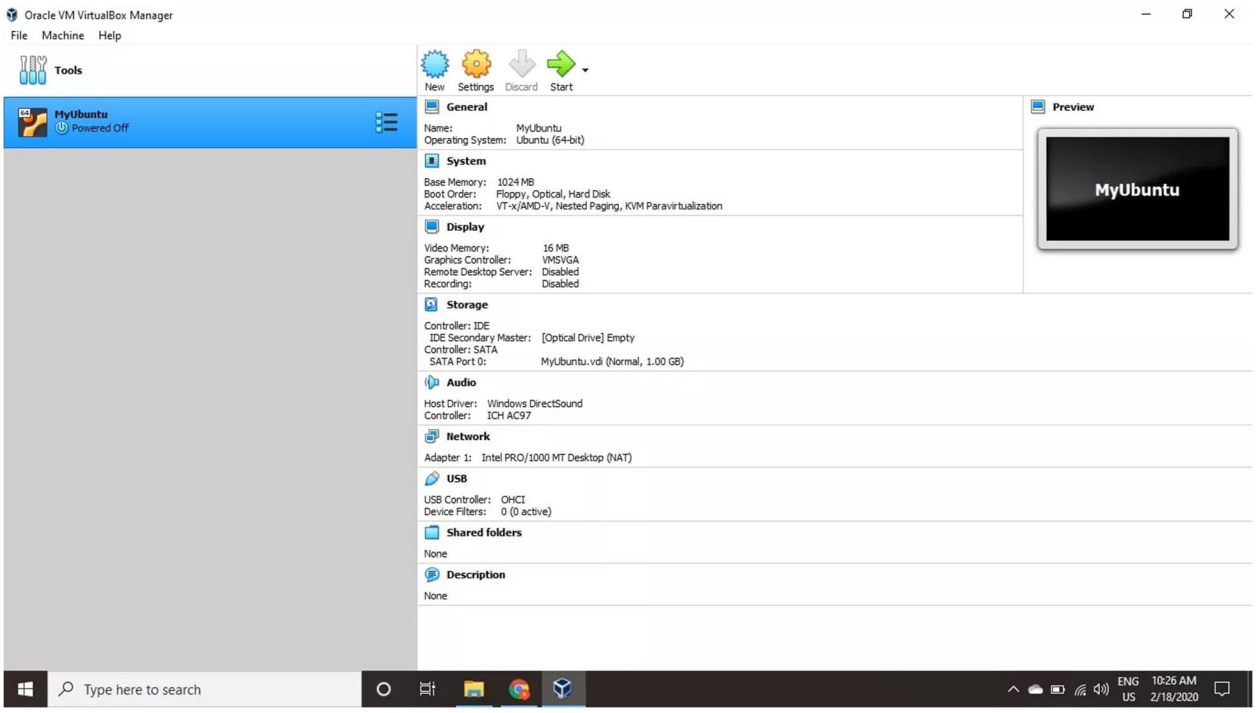

Oracle VM VirtualBox Manager
File Machine Help
Tools
Preferences Import Export New Add
Welcome to VirtualBox!
The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.
nation and latest news.
Create Virtual Machine
Memory size
Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.
The recommended memory size is 1024 MB.
1024 MB
4 MB
4096 MB
Next Cancel
Type here to search
ENG US 11:30 AM 2/17/2020

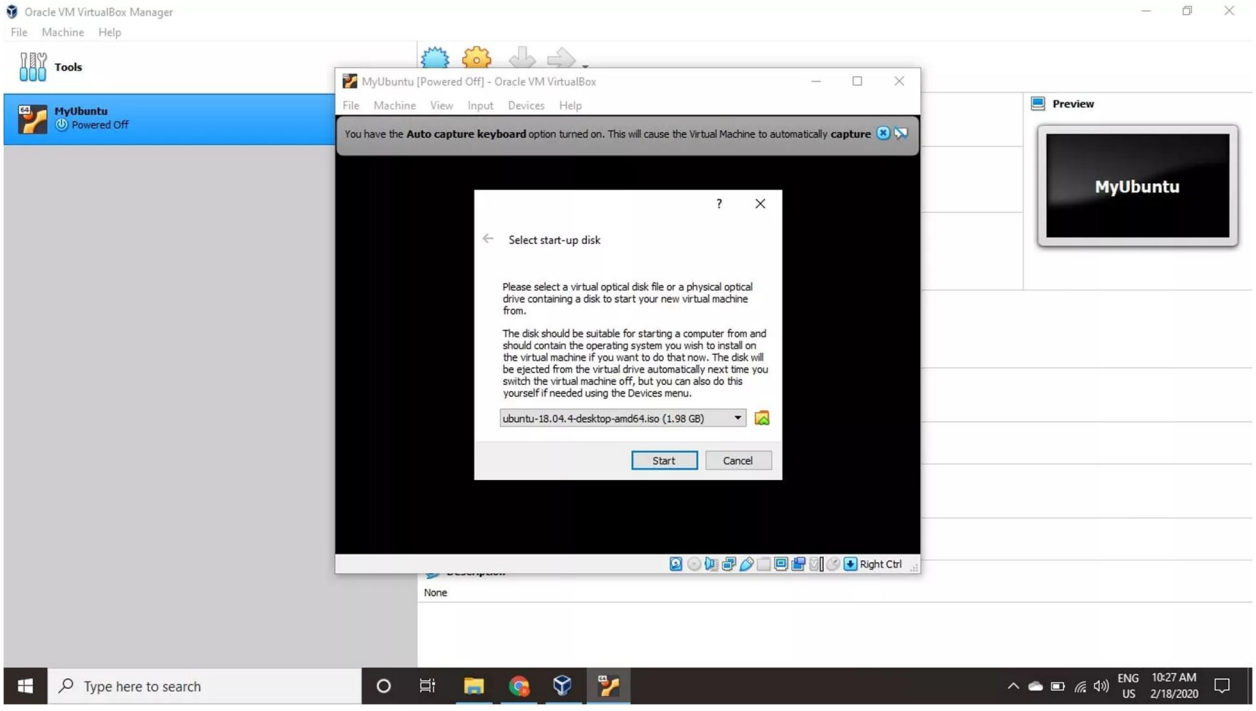

Oracle VM VirtualBox Manager
File Machine Help
Tools
Preferences Import Export New Add
Welcome to VirtualBox!
The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.
Create Virtual Machine
Hard disk
If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.
If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.
The recommended size of the hard disk is 10.00 GB.
Do not add a virtual hard disk
Create a virtual hard disk now
Use an existing virtual hard disk file
Empty
Create
Cancel
Type here to search
ENG US 11:37 AM 2/17/2020

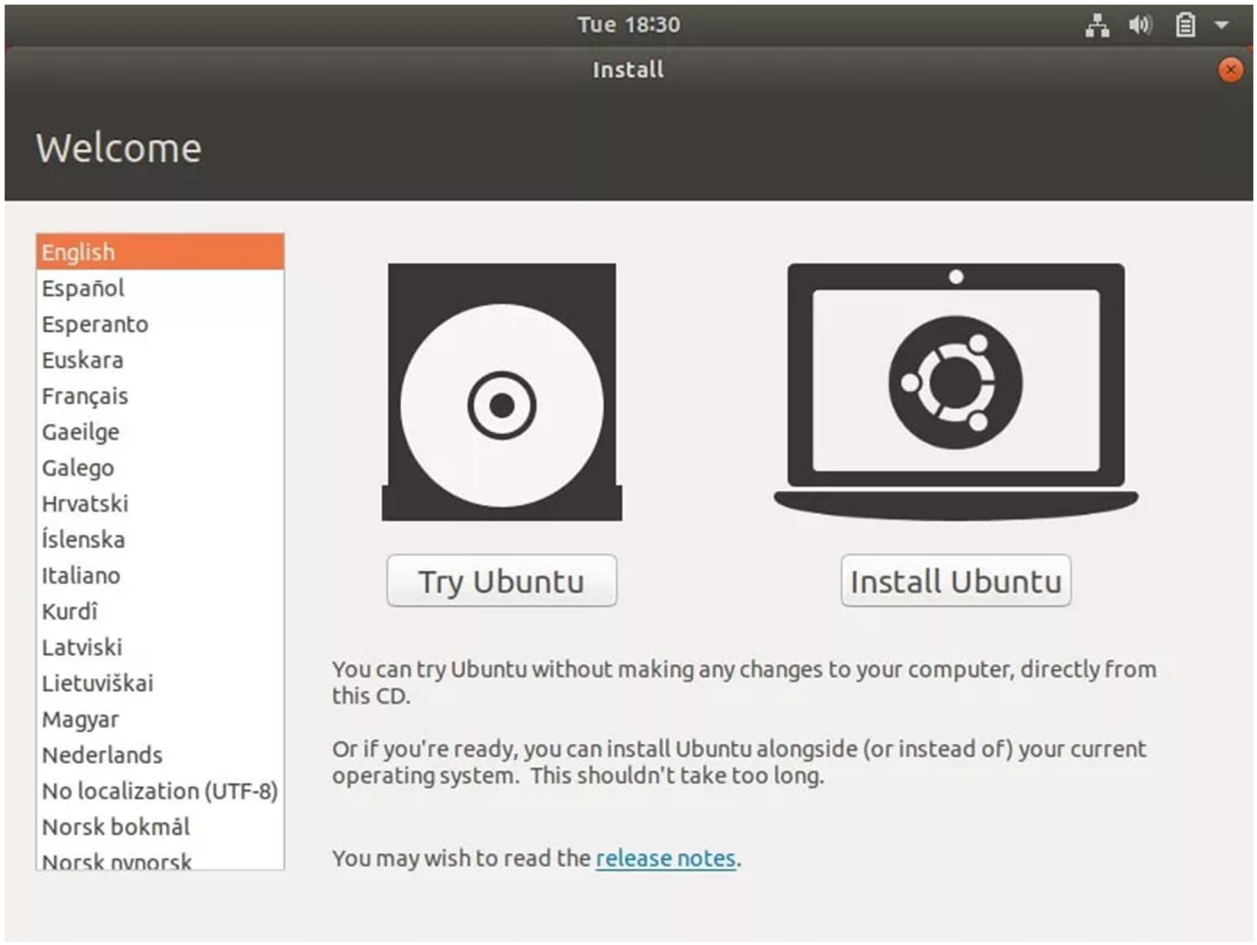

Oracle VM VirtualBox Manager
File Machine Help
Tools
Preferences Import Export New Add
Welcome to VirtualBox!
The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can
Create Virtual Hard Disk
Hard disk file type
Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.
VDI (VirtualBox Disk Image)
VHD (Virtual Hard Disk)
VMDK (Virtual Machine Disk)
Expert Mode
Next
Cancel
Type here to search
ENG US 11:39 AM 2/17/2020

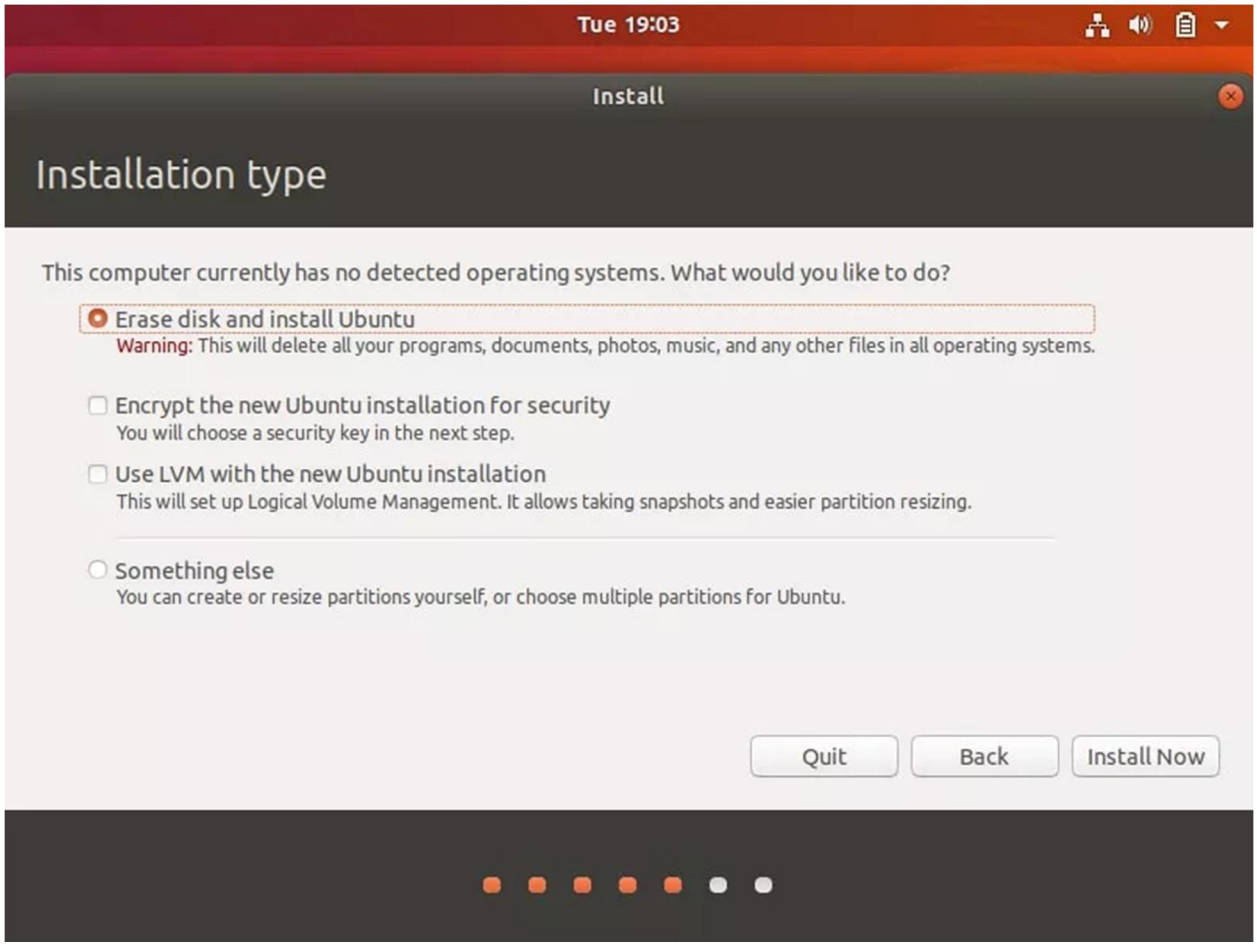

Oracle VM VirtualBox Manager
File Machine Help
Tools
Preferences Import Export New Add
Welcome to VirtualBox!
The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can
Create Virtual Hard Disk
Storage on physical hard disk
Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).
A dynamically allocated hard disk file will only use space on your physical hard disk as it fills up (up to a maximum fixed size), although it will not shrink again automatically when space on it is freed.
A fixed size hard disk file may take longer to create on some systems but is often faster to use.
Dynamically allocated
Fixed size
Next
Cancel
Type here to search
ENG INTL 7:59 PM 2/17/2020


Oracle VM VirtualBox Manager
File Machine Help
Tools
Preferences Import Export New Add
Welcome to VirtualBox!
The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can
Create Virtual Hard Disk
File location and size
Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.
C:\Users\Rob\VirtualBox VMs\MyUbuntu\MyUbuntu.vdi
Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.
10.00 GB
4.00 MB
2.00 TB
Create
Cancel
Type here to search
ENG US 10:14 AM 2/18/2020


Oracle VM VirtualBox Manager
File Machine Help
Tools
MyUbuntu
Powered Off
New Settings Discard Start
General
Name: MyUbuntu
Operating System: Ubuntu (64-bit)
System
Base Memory: 1024 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization
Display
Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled
Storage
Controller: IDE
IDE Secondary Master: [Optical Drive] Empty
Controller: SATA
SATA Port 0: MyUbuntu.vdi (Normal, 1.00 GB)
Audio
Host Driver: Windows DirectSound
Controller: ICH AC97
Network
Adapter 1: Intel PRO/1000 MT Desktop (NAT)
USB
USB Controller: OHCI
Device Filters: 0 (0 active)
Shared folders
None
Description
None
Preview
MyUbuntu
Type here to search
ENG US 10:26 AM 2/18/2020


Oracle VM VirtualBox Manager
File Machine Help
Tools
MyUbuntu
Powered Off
MyUbuntu [Powered Off] - Oracle VM VirtualBox
File Machine View Input Devices Help
You have the Auto capture keyboard option turned on. This will cause the Virtual Machine to automatically capture
Select start-up disk
Please select a virtual optical disk file or a physical optical drive containing a disk to start your new virtual machine from.
The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this yourself if needed using the Devices menu.
ubuntu-18.04.4-desktop-amd64.iso (1.98 GB)
Start
Cancel
Right Ctrl
None
Preview
MyUbuntu
Type here to search
ENG US 10:27 AM 2/18/2020


Tue 18:30
Install
Welcome
English
Español
Esperanto
Euskara
Français
Gaeilge
Galego
Hrvatski
Íslenska
Italiano
Kurdî
Latviski
Lietuviškai
Magyar
Nederlands
No localization (UTF-8)
Norsk bokmål
Norsk nynorsk
Try Ubuntu
Install Ubuntu
You can try Ubuntu without making any changes to your computer, directly from this CD.
Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.
You may wish to read the release notes.




# Lab 3: Shell Programming

**Introduction**

Knowing shell and practicing basics of shell programming in Ubuntu.

**Objective**

To familiarize yourself with shell programming.

**Tools/Software Requirement**

- Ubuntu/Terminal

**Tasks**

- Knowing what is Shell and terminal and their role.
- Knowing Ubuntu Editors
- Running first shell script

- Write a shell script to input your name and display this information on terminal.
- Write a shell script to copy multiple files into a directory.

**Implementation:**

- Write a shell script to input your name and display this information on terminal.

```
echo Enter your name
read name
echo Enter your program name
read prog
echo Enter your enrolement number
read enrol
clear
echo Details you entered
echo Name:$name
echo Program Name:$prog
echo Enrolement Number:$enrol
```

- Write a shell script to copy multiple files into a directory.

```
#! /bin/bash

outdir='/path/to/otherdir'
filename='filename.txt'

sourcedirs="dir1 dir2 dir3 dir4 dir5 dir6 dir7 dir7 dir9 dir10"

for d in $sourcedirs ; do
    cat "$d/$filename" >> "$outdir/$filename"
done
```

# Lab 4: Shell Programming

**Introduction**

Knowing shell and practicing basics of shell programming in Ubuntu.

**Objective**

To familiarize yourself with shell programming and enable the student to solve complex problems in Ubuntu using shell

**Tools/Software Requirement**

- Ubuntu/Terminal

**Tasks**

- Case…esac statement in shell
- While loop
- if statements
- if else statements

**Implementation:**

- Case statement

```
case  $variable-name  in
      pattern1)
          command1
          ...
          ....
          commandN
          ;;
      pattern2)
          command1
          ...
          ....
          commandN
          ;;
      patternN)
          command1
          ...
          ....
          commandN
          ;;
      *)
esac
```

```
# use case statement to make decision for rental
case $rental in
   "car") echo "For $rental rental is Rs.20 per k/m.";;
   "van") echo "For $rental rental is Rs.10 per k/m.";;
   "jeep") echo "For $rental rental is Rs.5 per k/m.";;
   "bicycle") echo "For $rental rental 20 paisa per k/m.";;
   "enfield") echo "For $rental rental Rs.3  per k/m.";;
   "thunderbird") echo "For $rental rental Rs.5 per k/m.";;
   *) echo "Sorry, I can not get a $rental rental  for you!";;
esac
```

- While loop

```
#!/bin/sh
INPUT_STRING=hello
while [ "$INPUT_STRING" != "bye" ]
do
echo "Please type something in (bye to quit)"
read INPUT_STRING
  echo "You typed: $INPUT_STRING"
done
```

- if statements and if else statements

```
#!/bin/sh
if [ "$X" -lt "0" ]
then
  echo "X is less than zero"
else
  echo "X is more than zero"
fi
```

## Lab 5: C Programming

**Introduction**

Introduce the student to write and compile code on Ubuntu (in C language)

**Objective**

To familiarize yourself with C programming in Ubuntu.

**Tools/Software Requirement**

- Ubuntu / gedit / gcc

**Tasks**

- Case…esac statement in shell
- While loop
- if statements
- if else statements

**Implementation:**

## Lab 6: C Programming for Fork

### Introduction

Introduce the student to write and compile code on Ubuntu (in C language) to practice creation of child process using fork. In this student will understand the scope of the processes and their termination.

### Objective

To familiarize yourself with creation of child processes in C programming in Ubuntu.

### Tools/Software Requirement

- Ubuntu / gedit / gcc

### Tasks

- fork() system call
- visualization of created processes using command line
- visualization of created processes using system monitor

### Implementation:

- fork() system call

```c
int main()
{
   pid_t pid;
   int x = 1;

   pid = Fork();
   if (pid == 0) {
      /* Child */
      printf("child : x=%d\n", ++x);
            exit(0);
   }

   /* Parent */
   printf("parent: x=%d\n", --x);
   exit(0);
}
```

*fork.c*

- visualization of created processes using command line

# Lab 7: C Programming for Pthread

## Introduction

Introduce the student to write and compile code on Ubuntu (in C language) to practice creation of child process using fork. In this student will understand the scope of the processes and their termination.

## Objective

To familiarize yourself with creation of child processes in C programming in Ubuntu.

## Tools/Software Requirement

- Ubuntu / gedit / gcc

## Tasks

- Creating and terminating Threads

Makefile | sum_on_many_threads.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

// The sum computed by the background thread
long long sum = 0;

// Thread function to generate sum of 0 to N
void* sum_runner(void* arg)
{
    long long *limit_ptr = (long long*) arg;
    long long limit = *limit_ptr;

    for (long long i = 0; i <= limit; i++) {
        sum += i;
    }

    // sum is a global variable, so other threads can access.
    pthread_exit(0);
}

int main(int argc, char **argv)
{
    if (argc < 2) {
        printf("Usage: %s <num 1> <num 2> ... <num-n>\n", argv[0]);
        exit(-1);
    }

    long long limit = atoll(argv[1]);

    // Launch thread
    pthread_attr_t attr;
    pthread_attr_init(&attr);
    pthread_t tid;
    pthread_create(&tid, &attr, sum_runner, &limit);

    // Wait until thread is done its work
```

```c
{
    long long *limit_ptr = (long long*) arg;
    long long limit = *limit_ptr;

    for (long long i = 0; i <= limit; i++) {
        sum += i;
    }

    // sum is a global variable, so other threads can access.
    pthread_exit(0);
}

int main(int argc, char **argv)
{
    if (argc < 2) {
        printf("Usage: %s <num 1> <num 2> ... <num-n>\n", argv[0]);
        exit(-1);
    }
    int num_args = argc - 1;

    long long limit = atoll(argv[1]);

    // Launch thread
    pthread_t tids[num_args];
    for (int i = 0; i < num_args; i++) {
        pthread_attr_t attr;
        pthread_attr_init(&attr);
        pthread_create(&tids[i], &attr, sum_runner, &limit);
    }

    // Wait until thread is done its work
    for (int i = 0; i < num_args; i++) {
        pthread_join(tids[i], NULL);
    }
    printf("Sum is %lld\n", sum);
}
```

- Compilation

**`gcc -lpthread -o hello myhello.c OR gcc -pthread -o hello myhello.c`**

- joins - Make a thread wait till others are complete (terminated).
- visualization of created threads using command line

# Lab 8: Inter-Process Communication

**Introduction**

Cooperative processes need some way of communication, Ubuntu provides IPC using pipes and filters.

**Objectives**

The purpose of this lesson is to introduce you to the way that you can construct powerful Unix command lines by combining Unix commands.

**Tools/Software Requirement**

- Ubuntu / gedit / gcc

**Description**

Conceptually, a pipe is a connection between two processes, used to implement IPC design amongst processes - where one or more processes would produce data and stream them on one end of the pipe, while other processes would consume the data stream from the other end of the pipe.

**Implementation:**

**IPC / Pipes.c**

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(){
FILE *read_fp;//**FILE
char buffer[BUFSIZ + 1];//**BUFSIZE
int chars_read;memset(buffer, '\0', sizeof(buffer));
read_fp = popen("uname-a", "r");//**fopen()
if (read_fp != NULL){
chars_read = fread(buffer, sizeof(char),BUFSIZ, read_fp);
if (chars_read > 0){
printf("Output was:-\n%s\n", buffer);}
pclose(read_fp);
exit(EXIT_SUCCESS);
}
exit(EXIT_FAILURE);
}
```

## Lab 9: Synchronization

### Introduction
Synchronization involves the orderly sharing of system resources by processes. System resource that is shared by two processes must be used in some defined order that do not leads towards race condition.

### Objectives

Objective is to understand synchronization.

### Tools/Software Requirement

- Ubuntu / gedit / gcc

### Description

Conceptually, a pipe is a connection between two processes, used to implement IPC design amongst processes - where one or more processes would produce data and stream them on one end of the pipe, while other processes would consume the data stream from the other end of the pipe.

### Implementation:

## Lab 10: CPU Scheduling (part 1)

### Introduction
Practice the famous algorithms of CPU scheduling.

### Objective
The Objective of this lab is to familiarize yourself with CPU scheduling

### Tools/Software Requirement

- Ubuntu / gedit / gcc

### Description:
To get insight of how the process manager handles removal of the running process from the CPU and the selection of another process on the basis of a particular strategy. The CPU scheduling is an essential part of a multitasking/multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

**Implementation:**

**Task # 1**

- Write a C program for FCFS CPU scheduling algorithm?

```
// C program for implementation of FCFS scheduling
#include<stdio.h>
// Function to find the waiting time for all processes
void findWaitingTime(int processes[], int n,
                     int bt[], int wt[])
{
   // waiting time for first process is 0
   wt[0] = 0;

   // calculating waiting time
   for (int  i = 1; i < n ; i++ )
      wt[i] =  bt[i-1] + wt[i-1] ;
}

// Function to calculate turn around time
void findTurnAroundTime( int processes[], int n,
              int bt[], int wt[], int tat[])
{
   // calculating turnaround time by adding
   // bt[i] + wt[i]
   for (int  i = 0; i < n ; i++)
      tat[i] = bt[i] + wt[i];
}

//Function to calculate average time
void findavgTime( int processes[], int n, int bt[])
{
   int wt[n], tat[n], total_wt = 0, total_tat = 0;

   //Function to find waiting time of all processes
   findWaitingTime(processes, n, bt, wt);

   //Function to find turn around time for all processes
   findTurnAroundTime(processes, n, bt, wt, tat);

   //Display processes along with all details
   printf("Processes   Burst time   Waiting time   Turn around time\n");

   // Calculate total waiting time and total turn
   // around time
   for (int  i=0; i<n; i++)
   {
```

```
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        printf("%d ",(i+1));
        printf("%d ", bt[i] );
        printf("%d",wt[i] );
        printf("%d\n",tat[i] );
    }
    int s=(float)total_wt / (float)n;
    int t=(float)total_tat / (float)n;
    printf("Average waiting time = %d",s);
    printf("\n");
    printf("Average turn around time = %d ",t);
}

// Driver code
int main()
{
    //process id's
    int processes[] = { 1, 2, 3};
    int n = sizeof processes / sizeof processes[0];

    //Burst time of all processes
    int  burst_time[] = {10, 5, 8};

    findavgTime(processes, n,  burst_time);
    return 0;
}
```

- Write a C program for round robin CPU scheduling algorithm?

```
#include <stdio.h>
// Function to calculate turn around time
int turnarroundtime(int processes[], int n,
int bt[], int wt[], int tat[]) {
  // calculating turnaround time by adding
  // bt[i] + wt[i]
  for (int i = 0; i < n ; i++)
  tat[i] = bt[i] + wt[i];
  return 1;
}
// Function to find the waiting time for all processes
int waitingtime(int processes[], int n,
int bt[], int wt[], int quantum) {
  // Make a copy of burst times bt[] to store remaining
  // burst times.
  int rem_bt[n];
  for (int i = 0 ; i < n ; i++)
  rem_bt[i] = bt[i];
```

```
  int t = 0; // Current time
  // Keep traversing processes in round robin manner
  // until all of them are not done.
  while (1) {
    bool done = true;
    // Traverse all processes one by one repeatedly
    for (int i = 0 ; i < n; i++) {
      // If burst time of a process is greater than 0
      // then only need to process further
      if (rem_bt[i] > 0) {
        done = false; // There is a pending process
        if (rem_bt[i] > quantum) {
          // Increase the value of t i.e. shows
          // how much time a process has been processed
          t += quantum;
          // Decrease the burst_time of current process
          // by quantum
          rem_bt[i] -= quantum;
        }
        // If burst time is smaller than or equal to
        // quantum. Last cycle for this process
        else {
          // Increase the value of t i.e. shows
          // how much time a process has been processed
          t = t + rem_bt[i];
          // Waiting time is current time minus time
          // used by this process
          wt[i] = t - bt[i];
          // As the process gets fully executed
          // make its remaining burst time = 0
          rem_bt[i] = 0;
        }
      }
    }
    // If all processes are done
    if (done == true)
      break;
  }
  return 1;
}
// Function to calculate average time
int findavgTime(int processes[], int n, int bt[],
int quantum) {
  int wt[n], tat[n], total_wt = 0, total_tat = 0;
  // Function to find waiting time of all processes
  waitingtime(processes, n, bt, wt, quantum);
  // Function to find turn around time for all processes
```

```
  turnarroundtime(processes, n, bt, wt, tat);
  // Display processes along with all details
  printf("Processes Burst Time Waiting Time turnaround time\n");
  // Calculate total waiting time and total turn
  // around time
  for (int i=0; i<n; i++) {
    total_wt = total_wt + wt[i];
    total_tat = total_tat + tat[i];
    printf("\t%d\t\t\t%d\t\t\t%d\t\t\t%d\n",i+1, bt[i], wt[i], tat[i]);
  }
  printf("Average waiting time = %f", (float)total_wt / (float)n);
  printf("\nAverage turnaround time = %f\n", (float)total_tat / (float)n);
  return 1;
}
// main function
int main() {
  // process id's
  int processes[] = { 1, 2, 3};
  int n = sizeof processes / sizeof processes[0];
  // Burst time of all processes
  int burst_time[] = {8, 6, 12};
  // Time quantum
  int quantum = 2;
  findavgTime(processes, n, burst_time, quantum);
  return 0;
}
```

## Lab 11 CPU scheduling (part 2)

### Introduction

Practice the basics algorithms of CPU scheduling.

### Objective

The Objective of this lab is to familiarize yourself with CPU scheduling and program basic algorithms related to it.

### Tools/Software Requirement

- Linux (Ubuntu)
- GCC

**Description:**

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy. Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

**Tasks**

- Write a C program for SJF (non-preemptive) CPU scheduling algorithm?

```
#include<stdio.h>
 int main()
{
   int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
   float avg_wt,avg_tat;
   printf("Enter number of process:");
   scanf("%d",&n);

   printf("nEnter Burst Time:n");
   for(i=0;i<n;i++)
   {
      printf("p%d:",i+1);
      scanf("%d",&bt[i]);
      p[i]=i+1;
   }

  //sorting of burst times
   for(i=0;i<n;i++)
   {
      pos=i;
      for(j=i+1;j<n;j++)
      {
         if(bt[j]<bt[pos])
             pos=j;
      }

      temp=bt[i];
      bt[i]=bt[pos];
      bt[pos]=temp;

      temp=p[i];
      p[i]=p[pos];
      p[pos]=temp;
   }

   wt[0]=0;
```

```
    for(i=1;i<n;i++)
    {
       wt[i]=0;
       for(j=0;j<i;j++)
          wt[i]+=bt[j];

       total+=wt[i];
    }

    avg_wt=(float)total/n;
    total=0;

    printf("nProcesst    Burst Time    tWaiting TimetTurnaround Time");
    for(i=0;i<n;i++)
    {
       tat[i]=bt[i]+wt[i];
       total+=tat[i];
       printf("np%dtt  %dtt    %dttt%d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=(float)total/n;
    printf("nnAverage Waiting Time=%f",avg_wt);
    printf("nAverage Turnaround Time=%fn",avg_tat);
}
```

- Write a C program for SRTF (Preemptive version of SJF) CPU scheduling algorithm?

```
#include <stdio.h>

int main()
{
    int arrival_time[10], burst_time[10], temp[10];
    int i, smallest, count = 0, time, limit;
    double wait_time = 0, turnaround_time = 0, end;
    float average_waiting_time, average_turnaround_time;
    printf("nEnter the Total Number of Processes:t");
    scanf("%d", &limit);
    printf("nEnter Details of %d Processesn", limit);
    for(i = 0; i < limit; i++)
    {
        printf("nEnter Arrival Time:t");
        scanf("%d", &arrival_time[i]);
        printf("Enter Burst Time:t");
        scanf("%d", &burst_time[i]);
        temp[i] = burst_time[i];
    }
    burst_time[9] = 9999;
    for(time = 0; count != limit; time++)
```

```
    {
        smallest = 9;
        for(i = 0; i < limit; i++)
        {
            if(arrival_time[i] <= time && burst_time[i] < burst_time[smallest] &&
burst_time[i] > 0)
            {
                smallest = i;
            }
        }
        burst_time[smallest]--;
        if(burst_time[smallest] == 0)
        {
            count++;
            end = time + 1;
            wait_time = wait_time + end - arrival_time[smallest] - temp[smallest];
            turnaround_time = turnaround_time + end - arrival_time[smallest];
        }
    }
    average_waiting_time = wait_time / limit;
    average_turnaround_time = turnaround_time / limit;
    printf("nnAverage Waiting Time:t%lfn", average_waiting_time);
    printf("Average Turnaround Time:t%lfn", average_turnaround_time);
    return 0;
}
```

## Lab 10: Address Translation

### Introduction

To manage primary memory. Memory management functionality keeps track of each and every memory location either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes.

### Objective

To understand the concepts of memory management.

### Tools/Software Requirement

- Linux (Ubuntu)
- GCC

**Task**

Write a C program to implement memory management using segmentation.

```
#include<stdio.h>
#include<conio.h>
struct list
{
int seg;
int base;
int limit;
struct list *next;
} *p;
void insert(struct list *q,int base,int limit,int seg)
{
if(p==NULL)
{
p=malloc(sizeof(Struct list));
p->limit=limit;
p->base=base;
p->seg=seg;
p->next=NULL;
}
else
{
while(q->next!=NULL)
{
Q=q->next;
Printf("yes")
}
```

```
q->next=malloc(sizeof(Struct list));
q->next ->limit=limit;
q->next ->base=base;
q->next ->seg=seg;
q->next ->next=NULL;
}
}
int find(struct list *q,int seg)
{
while(q->seg!=seg)
{
q=q->next;
}
return q->limit;
}
int search(struct list *q,int seg)
{
while(q->seg!=seg)
{
q=q->next;
}
return q->base;
}
main()
{
p=NULL;
int seg,offset,limit,base,c,s,physical;
printf("Enter segment table/n");
```

```
printf(“Enter -1 as segment value for termination\n”);
do
{
printf(“Enter segment number”);
scanf(“%d”,&seg);
if(seg!=-1)
{
printf(“Enter base value:”);
scanf(“%d”,&base);
printf(“Enter value for limit:”);
scanf(“%d”,&limit);
insert(p,base,lmit,seg);
}
}
while(seg!=-1)
printf(“Enter offset:”);
scanf(“%d”,&offset);
printf(“Enter bsegmentation number:”);
scanf(“%d”,&seg);
c=find(p,seg);
s=search(p,seg);
if(offset<c)
{
physical=s+offset;
printf(“Address in physical memory %d\n”,physical);
}
else
{
```

```
printf(“error”);
}
}
```