Great Human Compiler!

SNU CSE 4190.308 Lab #2

Sungmin Kim

Sangjin Ha

Architecture & Code optimization (ARC) Lab

October 21st, 2016

Overview

- **The goal of this lab**
  - To make *get_min_max()* function work correctly
- **By finishing this lab successfully**
  - You will understand how a function call works in x86-64.
  - You will prove you're a great human compiler, indeed. :-)
- **You need a Linux environment to do this lab**
  - You can access a Linux machine in "Software Lab" at Building 302 (Room 311-1) with your own ID and password.
  - You can also install Linux on your PC using VirtualBox. (See Appendix B)

Problem Specifications (1)

- **You are given an incomplete program: *get_min_max***
  - Input: *n* integers
  - Output: maximum and minimum values among *n* integers
- **Your task: Complete the *get_min_max()* function in get_min_max.s file using x86-64 assembly!**
  - Function signature

```
void get_min_max (int arr[], int n, int* min, int* max);
```

`int arr[]`	an array for input integers
`int n`	the number of input integers
`int* min, max`	pointers to the variables which holds minimum and maximum values

Problem Specifications (2)

- **Restrictions**
  - You **CANNOT** use **%*cx and %*dx** registers in your code.
  - You **CANNOT** modify the first & last 3 instructions (`mov`, `push`, `pop`, & `ret`) in *get_min_max.s.*
  - You can only modify the *write-your-code-here* section in *get_min_max.s.*

```
(get_min_max.s)

  #-------------------------------------------------
  # Do NOT use %*cx and %*dx registers
  #-------------------------------------------------

  # Write your codes here

  #-------------------------------------------------
```

Problem Specifications (3)

- **Skeleton codes**
  - `Makefile`: file for GNU `make` utility
  - `main.c`: C program which takes *n* integers for input array and prints the minimum/maximum values by calling your `get_min_max()`
  - `get_min_max.s`: body of `get_min_max()` you should implement
- **You can simply invoke "`make`" in your shell to compile these modules and build an executable `get_min_max`.**

Problem Specifications (4)

- **Example**

  a. Compile your program using command `make`

  b. Execute `get_min_max`

  c. Enter the number of integers (*n*) and *n* input integers

  d. Verify that min and max values are right

```
$ make
gcc -g -O -w -c main.c -o main.o
as  get_min_max.s -o get_min_max.o
gcc  main.o get_min_max.o -o get_min_max
$ ./get_min_max
Enter the number of integers: 5
Enter the integers: 4190 308 10 -21 2016
min: -21, max: 4190
$
```

Submission Guideline

- **Rename the file name of** `get_min_max.s` **to** "`YourStudentID.s`"
- **Send us e-mail (snu.comarch.2016@gmail.com) with** `YourStudentID.s` **as attachment.**
- **Subject of e-mail: "[CA Lab2] YourStudentID"**
- **Due date is Nov 7th (Mon) 11:59 AM (after the midterm).**

Grading Policy

- **Working correctly: 100 point**
  - We will use various inputs to verify your code
- **Penalty for violating restriction**
  - Using ***cx** register: -20%
  - Using ***dx** register: -20%
  - Code outside the *write-your-code-here* section: -20%
- **Penalty for late submission: -20 % per every 24 hours**

Troubleshooting

- ***make* command doesn't work**
  - Problem 1: `$> make: command not found`
  - Solution 1: `$> sudo apt-get install build-essential`

  - Problem 2: `$> make: *** No targets specified and no makefile found.  Stop.`
  - Solution 2: `$> cd` *`/your/path/to/lab2/`*
    - e.g. `$> cd ~/Downloads/Lab2`

Q&A

Appendix A

- How to use GNU debugger

How to use GNU debugger

- **Using GNU debugger (=GDB)**
  - You can see what is going on inside a program while it is running
  - You can start your program, specifying anything that might affect its behavior
  - You can make your program stop under a specified condition
  - You can examine what has happened (i.e., the program's state), when your program has stopped
  - Can change the value of variable in your program

How to use GNU debugger

- **Install GDB**

```
$> sudo apt-get install gdb
```

- **Run executable with GDB**

```
$> gdb nameOfExecutable
```

  - In this lab, “nameOfExecutable” is ‘**bomb**’

How to use GNU debugger

- **Basic instructions**
  - (gdb) `run` : Start the program
  - (gdb) `continue` : Run the program until next breakpoint
  - (gdb) `breakpoint` : Make a breakpoint
  - (gdb) `delete` : Delete a breakpoint
  - (gdb) `step` : Run next line of code
  - (gdb) `next` : Run next line of code (not jumping into a function)
  - (gdb) `quit` : Quit gdb
- **Instructions for assembly code**
  - (gdb) `disassemble` : Disassemble the function / lines of code
  - (gdb) `stepi` : Run next line of assembly code

How to use GNU debugger

- **Variable print instruction**
  - (gdb) `print` *func* : Print address of function *func*
  - (gdb) `p` *var* : Print value of variable *var*
  - (gdb) `p/[format]` *var* : Print value of *var* with format
    - Format: t = binany, o = octal, d = int, u = unsigned int, x = hexadecimal, c = char, f = floating-point
- **Memory print instruction**
  - (gdb) `x/[range][format][unit]` *addr* : Print memory value
    - Format: t = binany, o = octal, d = int, u = unsigned int, x = hexadecimal, c = char, f = floating-point, **s = string**, **i = assembly instr**
    - Unit: b = byte, h = halfword (2-byte), w = word (4-byte), g = giant word (8-byte)

How to use GNU debugger

- **Information print instruction**
  - (gdb) `info registers` : Print all registers' value
  - (gdb) `info breakpoints` : Print all breakpoints

How to use GNU debugger

- **If you need more instruction detail**

```
$> man gdb
```

  - (gdb) help

- **References**
  - **http://visualgdb.com/gdbreference/commands/**
  - http://www.yolinux.com/TUTORIALS/GDB-Commands.html
  - 유닉스 리눅스 프로그래밍 필수 유틸리티, 백창우, 한빛미디어

Appendix B

- How to set up a Linux environment

Option 1: Windows' bash shell

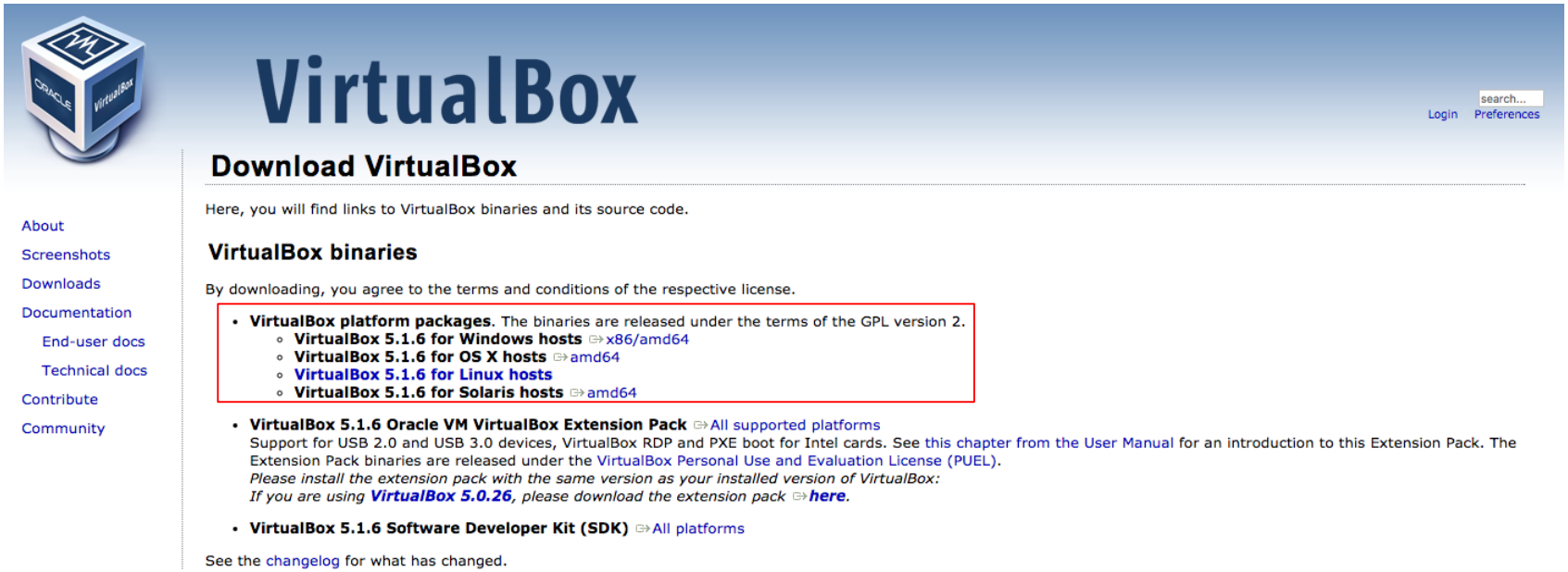
- **It can run Linux command-line utilities on Windows 10 (64bit only)**
- **It's based on Ubuntu**

- **Here is a setup guideline**
  - http://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/
- **Get utilities for the lab**

```
$> sudo apt-get install build-essential gdb
```

Option 2: Ubuntu on VirtualBox

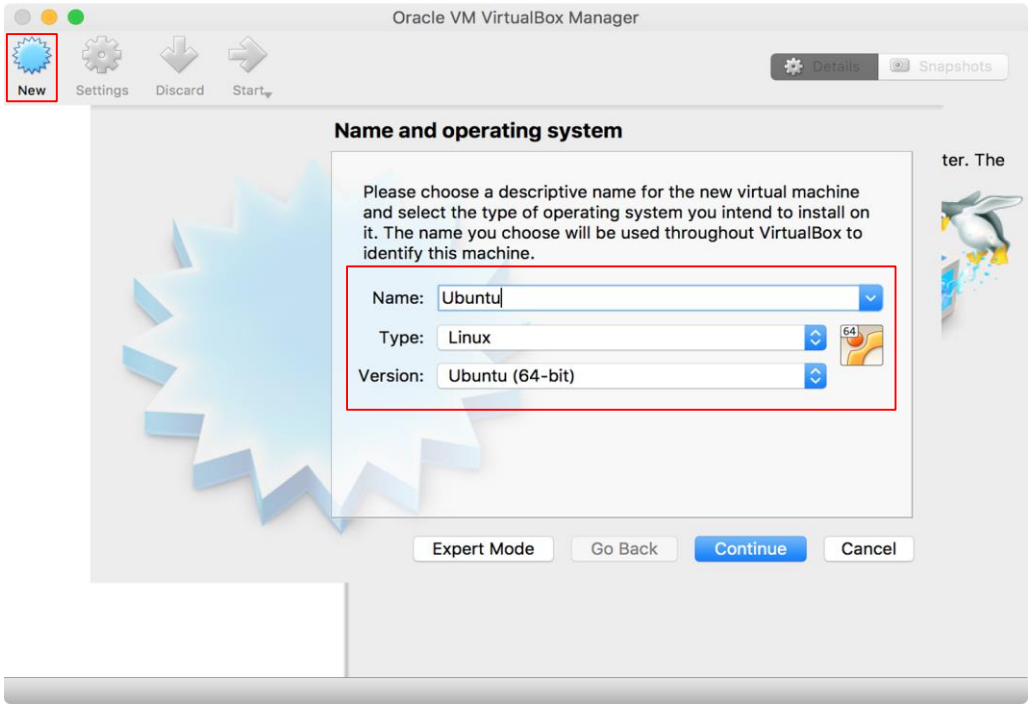
- Download and install Oracle VirtualBox 5.1



- Get Ubuntu 14.04 LTS CD image
  - http://releases.ubuntu.com/14.04/ubuntu-14.04.4-desktop-amd64.iso

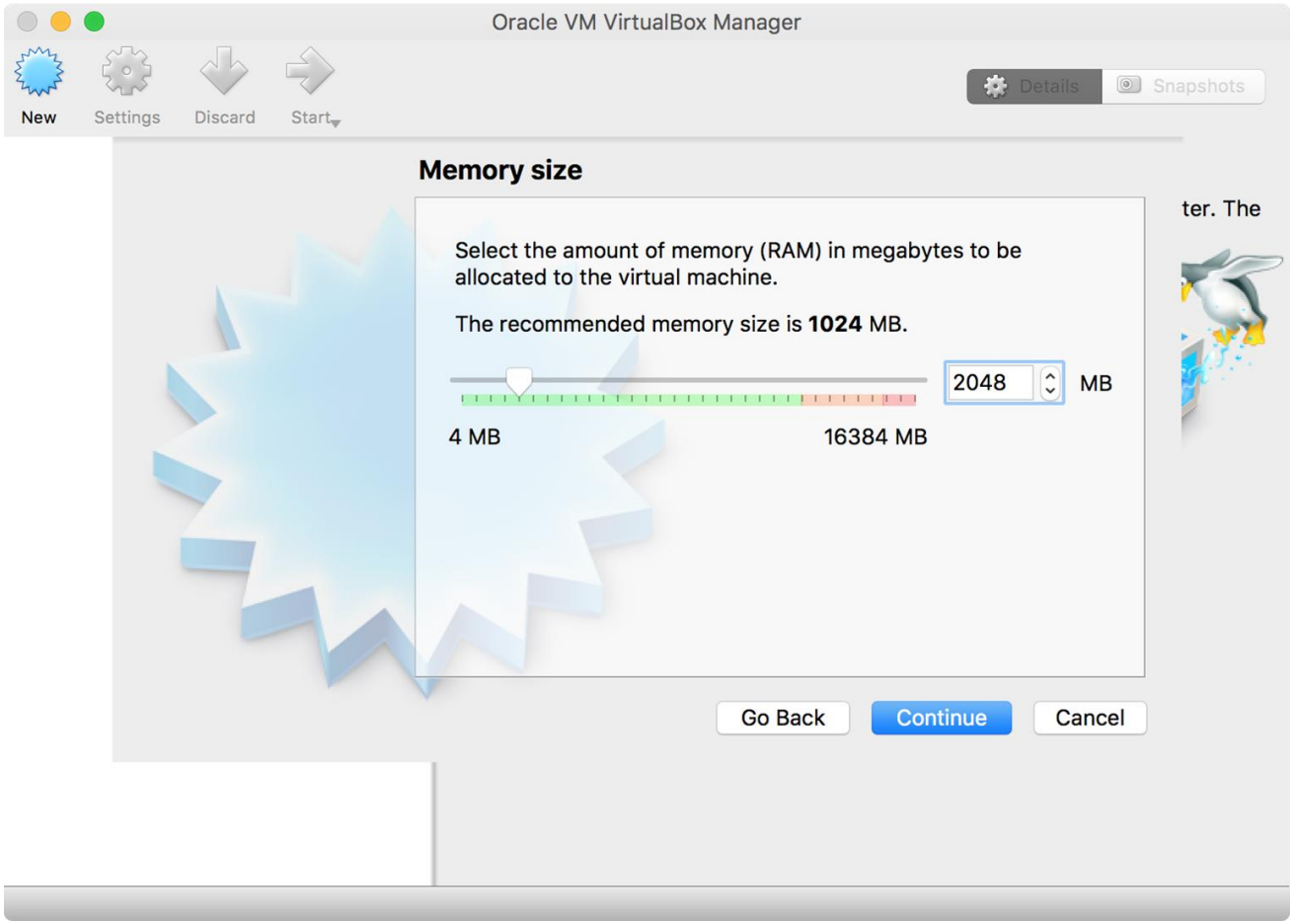
Install Ubuntu on VirtualBox

- Run VirtualBox
- Click “New”
- Type “Ubuntu” into name
- Make sure that the OS type and version are correct



Install Ubuntu on VirtualBox

- **The recommended memory size depends on your system (default is 1GB)**

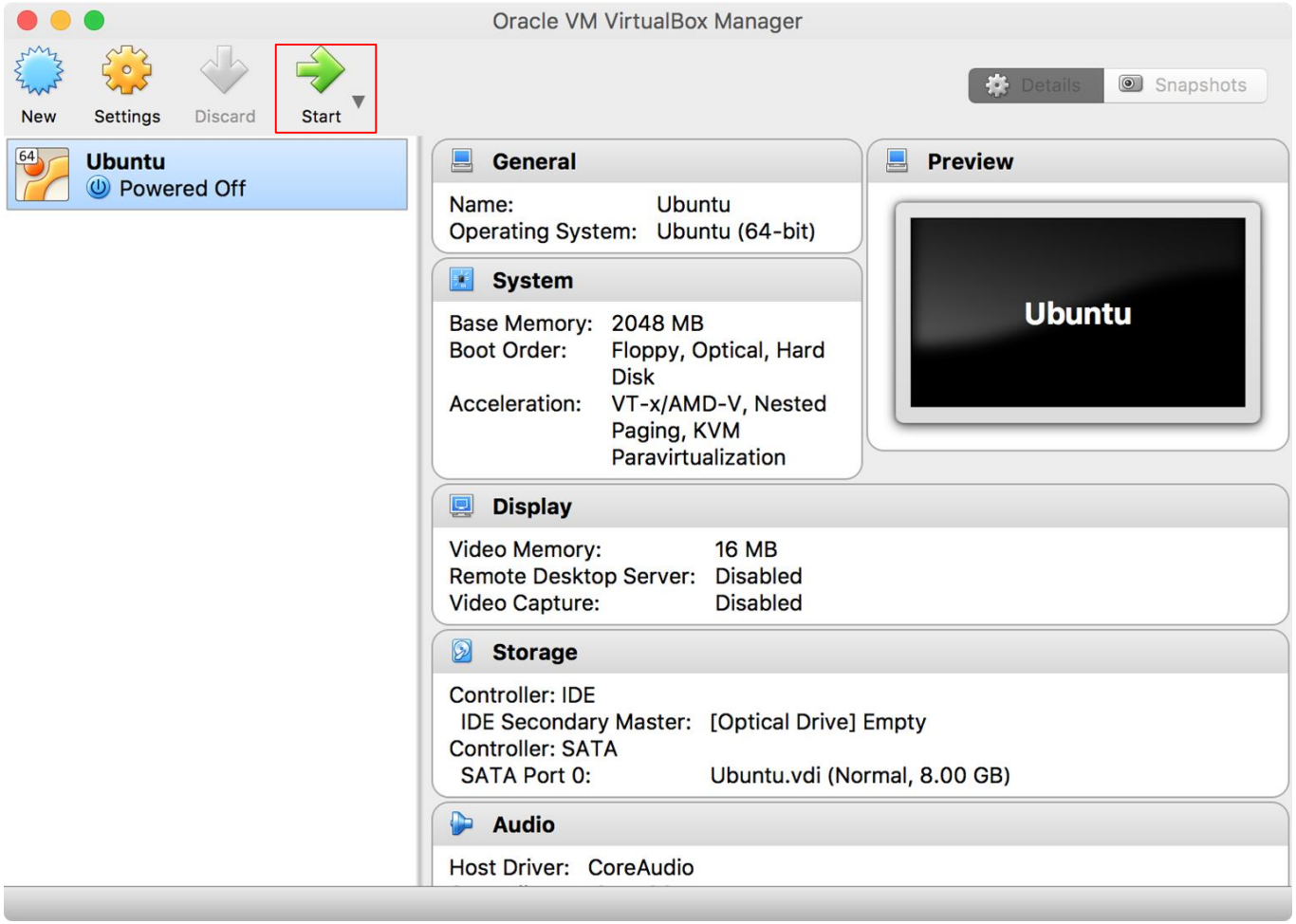


Install Ubuntu on VirtualBox

- "Create, Continue, Continue, Create"

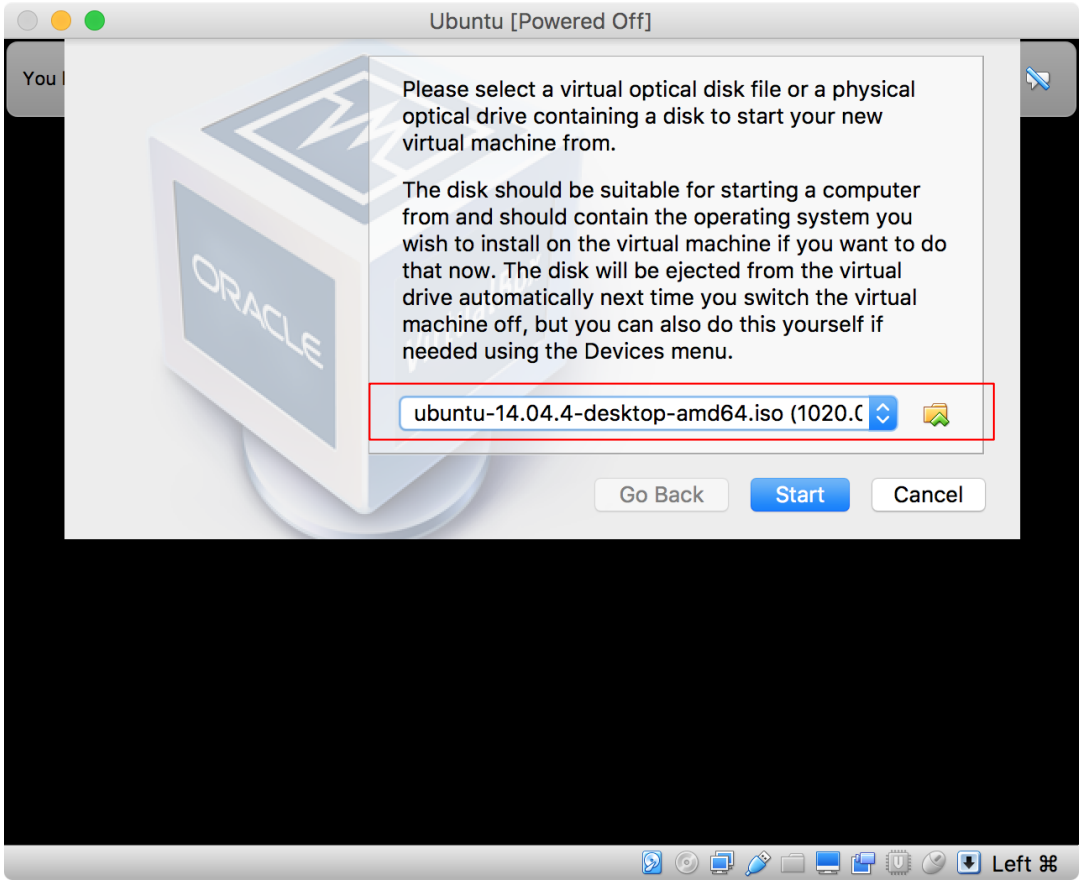
Install Ubuntu on VirtualBox

- **Now, click "Start"**



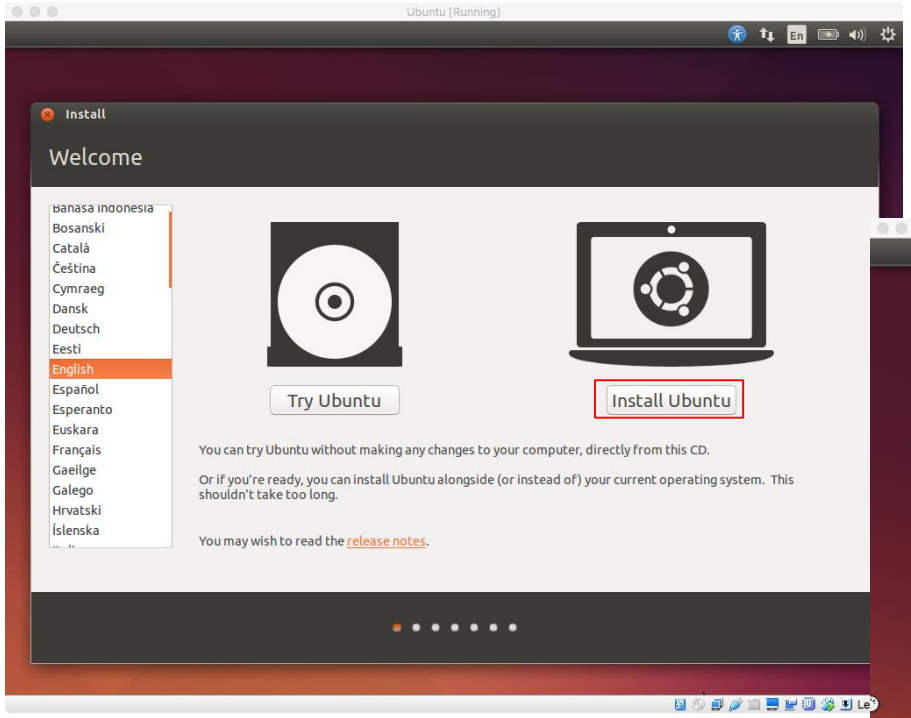
Install Ubuntu on VirtualBox

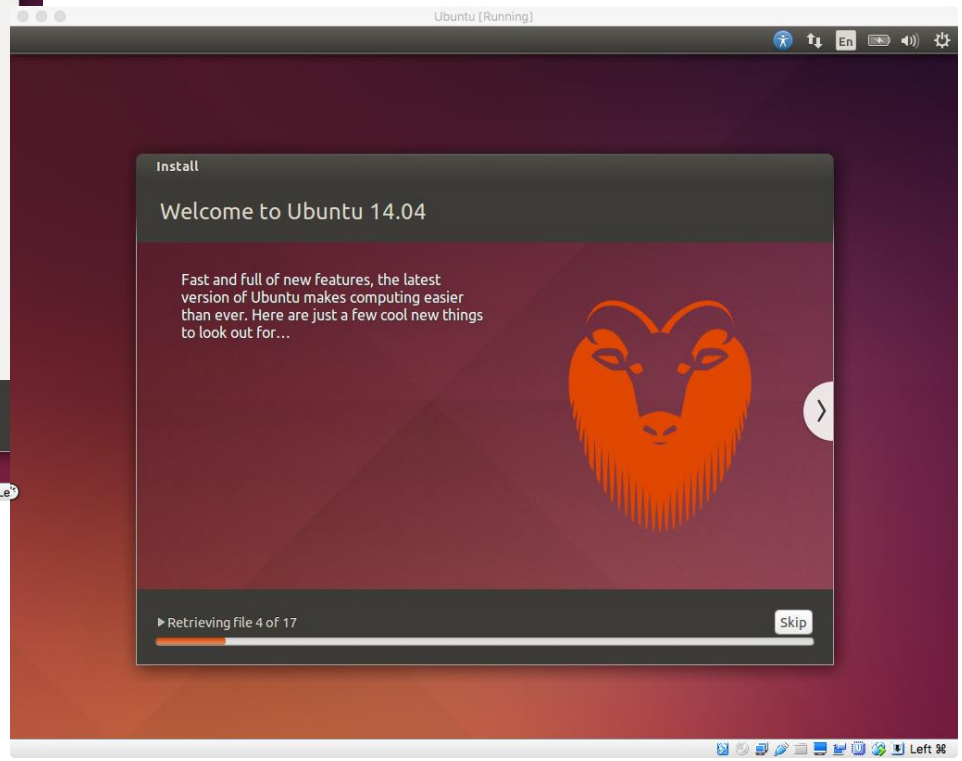
- Browse the downloaded Ubuntu 14.04 ISO image
- Click “Start” to boot



Install Ubuntu on VirtualBox

- **Install Ubuntu and now you are ready to go!**





(Optional) VirtualBox Guest Addition

- Automatic adjustment of resolution of guest OS
- Integration of mouse and keyboard