# Setup on Personal Machine

**Install either your favorite version of Linux to your personal machine (If new to Linux, Ubuntu recommended)**

**Or Install your favorite virtual machine software (If new to virtualization/don't want to hassle with paying or registering, VirtualBox recommended) to run Linux in a virtual machine**

**To set up environment, at terminal prompt:**

**$: sudo apt-get install qemu build-essential nasm gcc-multilib git**

**To get geekOS code:**

**$: git clone git://scriptroute.cs.umd.edu/geekos-project**

**Before running geekOS, change “qemu” to “qemu-system-i386” on the following lines of geekos-project/build/Makefile: 5, 12, 15.**

**Also, if your personal machine is an i386, then change “/usr/bin/qemu” on line 15 in geekos-project/build/Makefile.linux to “/usr/bin/qemu-system-i386”. Otherwise, if your machine is an x86_64, change “/usr/bin/qemu” to “/usr/bin/qemu-system-i386” on line 12 of geekos-project/build/Makefile.linux.x86_64**

**To run geekOS:**

**$: cd geekos-project/build**

**$: make run**

# Setup On LinuxLab

- **Retrieve username/password for CMSC412 on grades.cs.umd.edu**
  - **username is of the form 412xxx (x is a digit)**
  - **Password is of the form (Word)(StringOfDigits)(Word)**
- **From terminal prompt on your computer or using ssh program from Windows:**

  **$: ssh username@linuxlab.csic.umd.edu -X (make sure to enable X11 forwarding when using ssh program with Windows)**
  - **Enter password provided from grades.cs.umd.edu**
- **At linuxlab terminal prompt:**

  **$: git clone git://scriptroute.cs.umd.edu/geekos-project**

  **$: cd geekos-project/build**

  **$: make run**

# geekOS Debugging

- From a terminal prompt:
  - $: cd geekos-project/build
  - $: make dbgrun
- Open new terminal:
  - $: cd geekos-project/build
  - $: make dbg
- At dbg prompt:
  - $: “continue” (or “c” to run)
  - Set breakpoints as described in assignment handout

# struct File

- Important struct that contains most information for a File, will be used throughout course
- Located in geekos-project/include/geekos/vfs.h
- struct File_Ops *ops is used to control which operations can be performed on File.  See struct File_Ops
- ulong_t filePos and ulong_t endPos denote current position in and end position of file, respectively
- void *fsData will be used for several purposes, specifically in this project will point to Pipe.  See project0 assignment handout, Figure 1
- int mode will determine mode of file
- struct Mount_Point * mountPoint is used to denote which filesystem the File is a mounted on.  See struct Mount_Point

# geekos-project/src/geekos/syscall.c

- **You will become very familiar with this file, almost all of the projects will involve editing this file**
- **You can see system calls for other projects by looking for TODO_P() statements**
- **geekOS will arrive here from user space when a system call is made. For instance, in project0, geekos-project/src/user/pipe-p1.c:**
  - **1) In user space, line 15, calls Pipe()**
  - **2) Syscall_Handler() in geekos-project/src/geekos/trap.c will handle the Pipe system call**
  - **3) Control is then handed to Sys_Pipe() in geekos-project/src/geekos/syscall.c through line 58 in geekos-project/src/geekos/trap.c:**

    **state->eax = g_syscallTable[syscallNum] (state);**

# geekos-project/src/geekos/pipe.c

- Four functions you will need to implement.
- Pipe_Create() will initialize everything that is needed for a Pipe to operate.  See section 3.4 of project0 assignment handout, especially Figure 1
- Pipe_Read(), Pipe_Write(), and Pipe_Close() will be called when a Read(), Write(), or Close() is performed on the appropriate file descriptor in geekos-project/src/user/pipe-p1.c and geekos-project/src/user/pipe-p2.c
- struct File_Ops Pipe_Read_Ops and struct File_Ops Pipe_Write_Ops will allow Pipe_Read(), Pipe_Write(), and Pipe_Close() to be called

# Getting started

- To get familiar with the geekOS code and see any change in your project, try to get a message to print out when Pipe() is called, so when you run pipe-p1.exe or pipe-p2.exe in qemu you see that message before TODO_P() is called.
- In geekos-project/src/geekos/syscall.c:
  - Put the following line as the first statement in the Sys_Pipe() function:
    - Print("You have just called pipe\n");
- Open a new terminal:

  $: cd geekos-project/build

  $: make run
- In qemu prompt:

  $: pipe-p1
- It is encouraged to use the debugger to set breakpoints throughout geekos-project/src/geekos/syscall.c and geekos-project/src/geekos/pipe.c to get more familiar with the code base