CS 1550

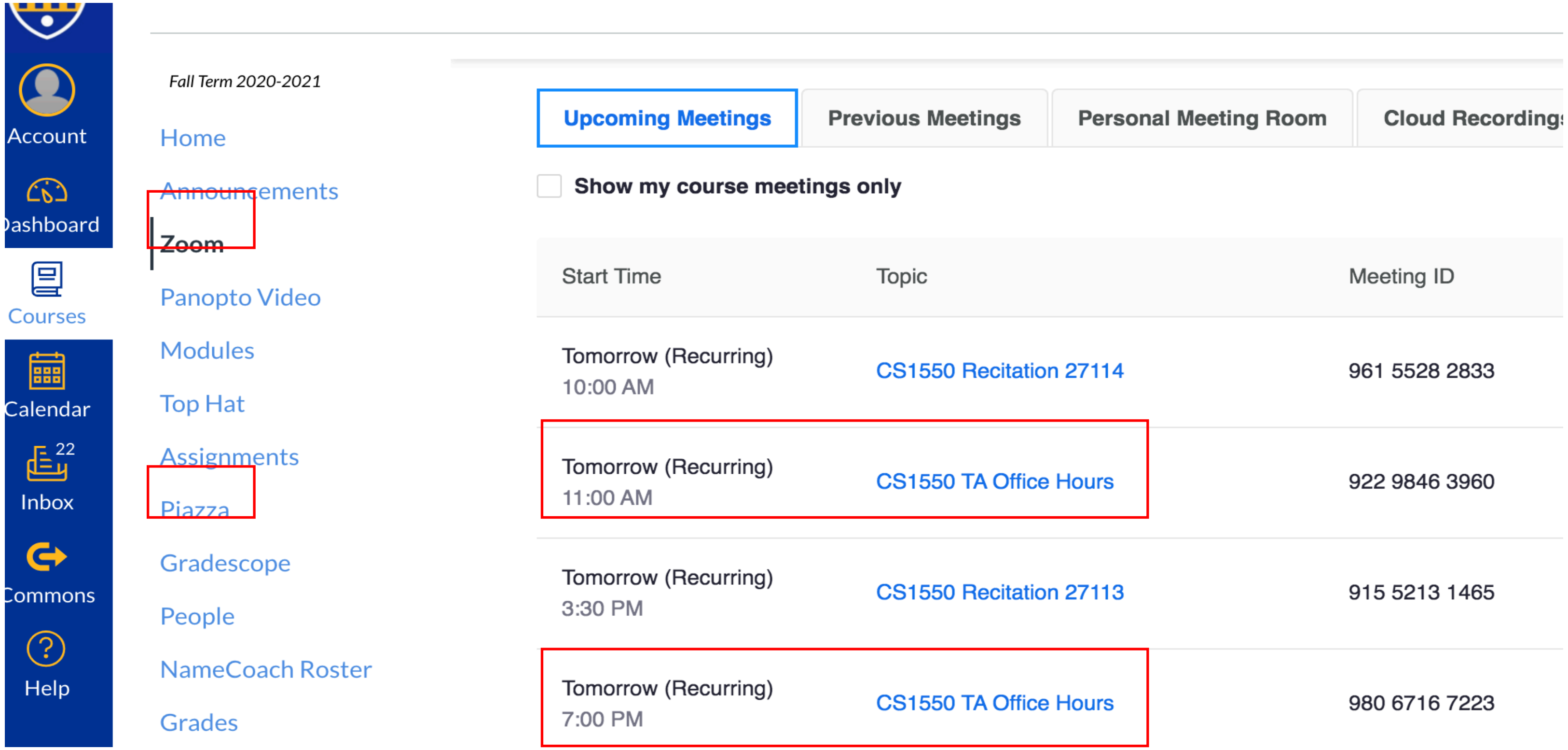
Introduction and Lab 1

Jinpeng Zhou

jiz150@pitt.edu

Office Hours

- Office Hours (Zoom): Friday 11AM-3PM, and 7pm-9pm
- Email: jiz150@pitt.edu
- Piazza



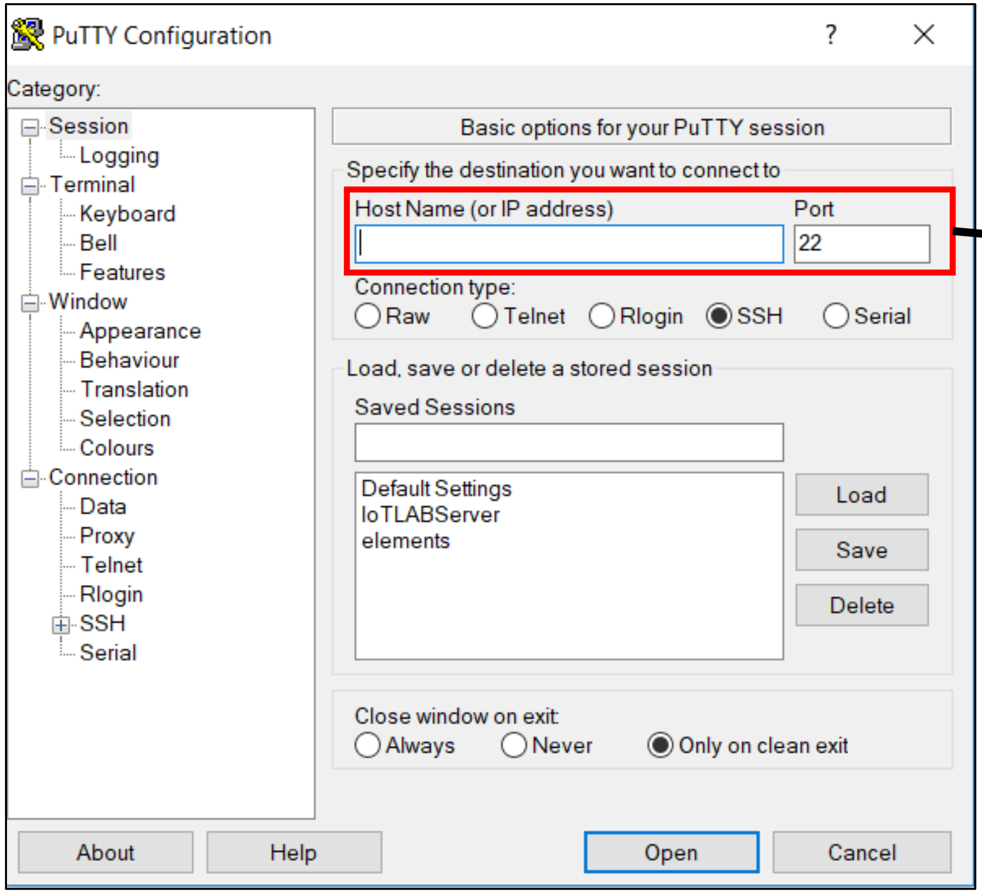
Start Time	Topic	Meeting ID
Tomorrow (Recurring) 10:00 AM	CS1550 Recitation 27114	961 5528 2833
Tomorrow (Recurring) 11:00 AM	CS1550 TA Office Hours	922 9846 3960
Tomorrow (Recurring) 3:30 PM	CS1550 Recitation 27113	915 5213 1465
Tomorrow (Recurring) 7:00 PM	CS1550 TA Office Hours	980 6716 7223

Use SSH to log into server

- Windows
  - **Putty**
- MacOS/Ubuntu
  - **Terminal**

Windows - Putty

- Download from **www.putty.org**



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

Connection type:
○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

Saved Sessions

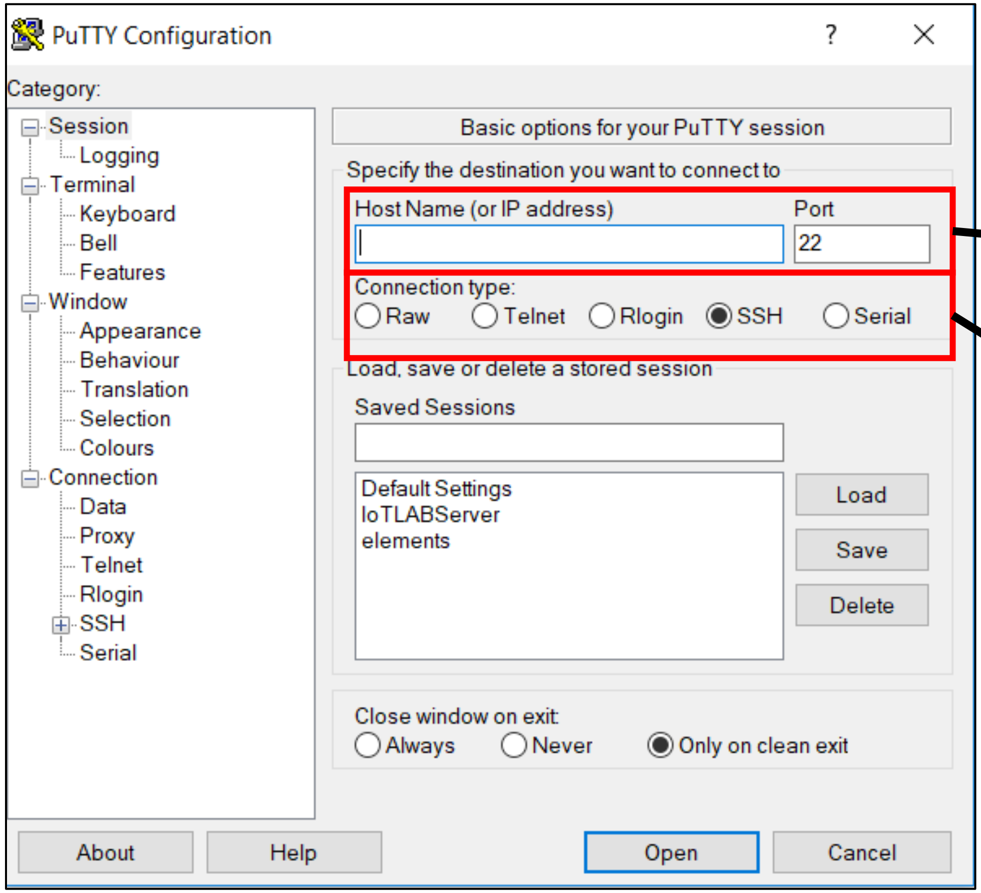
Default Settings
IoTLABServer
elements

Close window on exit:
○ Always ○ Never ● Only on clean exit

server address:
linux.cs.pitt.edu at port 22

Windows - Putty

- Download from **www.putty.org**



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

Connection type:
○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

Saved Sessions

Default Settings
IoTLABServer
elements

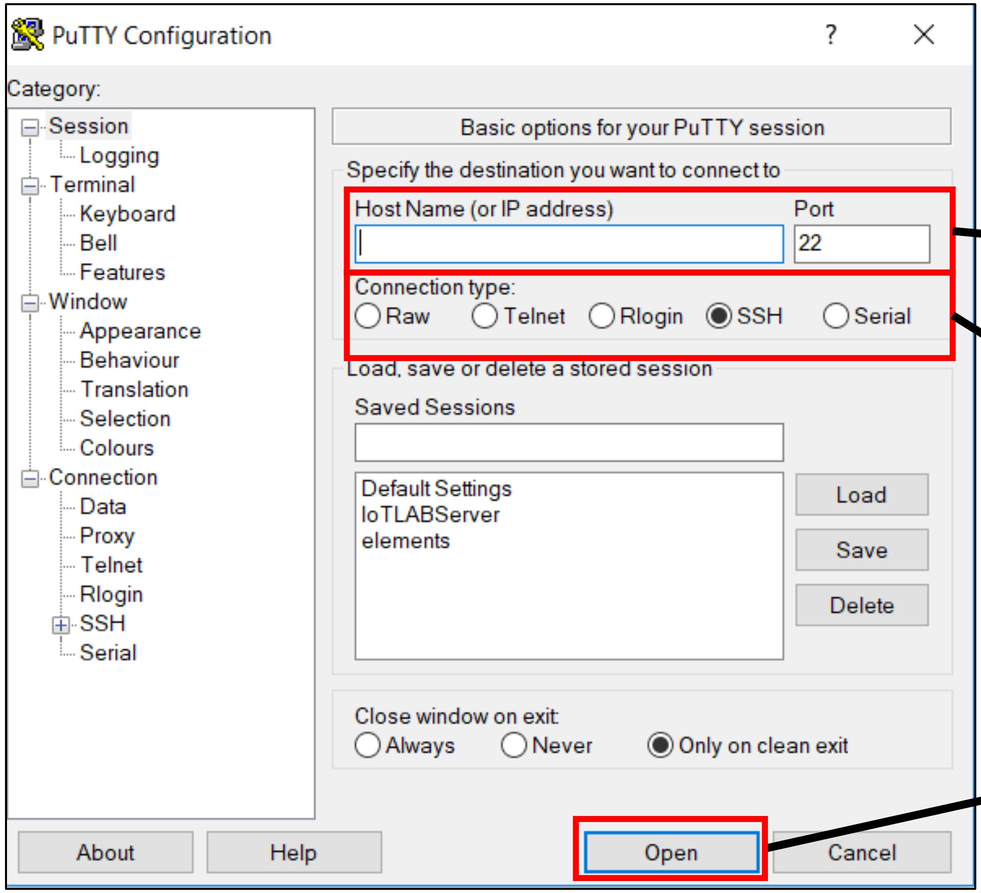
Close window on exit:
○ Always ○ Never ● Only on clean exit

server address:
linux.cs.pitt.edu at port 22

Use SSH

Windows - Putty

- Download from **www.putty.org**



Specify the destination you want to connect to

Host Name (or IP address)

Port 22

Connection type:

○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Saved Sessions

Default Settings
IoTLABServer
elements

Close window on exit:

○ Always ○ Never ● Only on clean exit

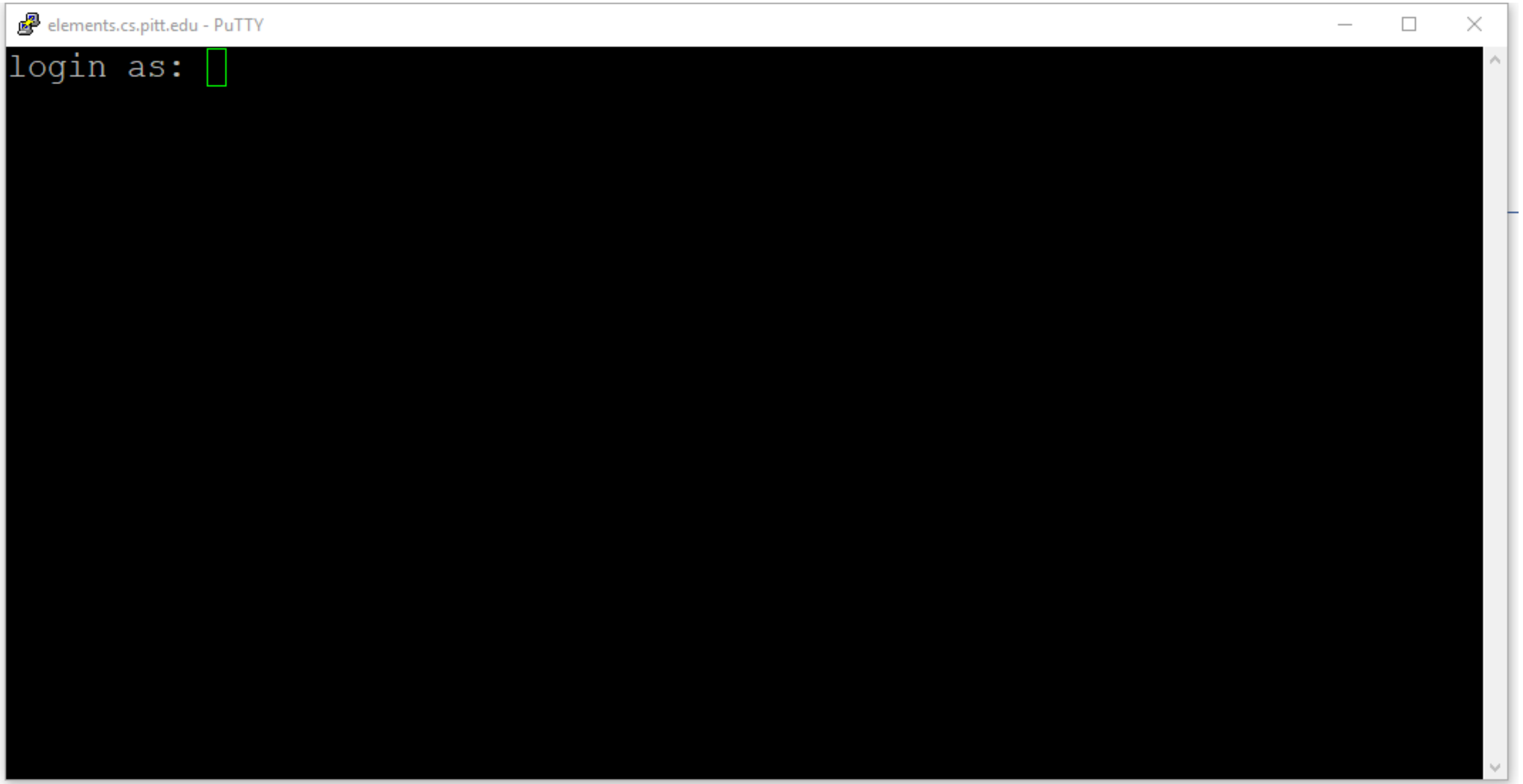
server address, e.g:
linux.cs.pitt.edu at port 22

Use SSH

Click Open

Windows - Putty

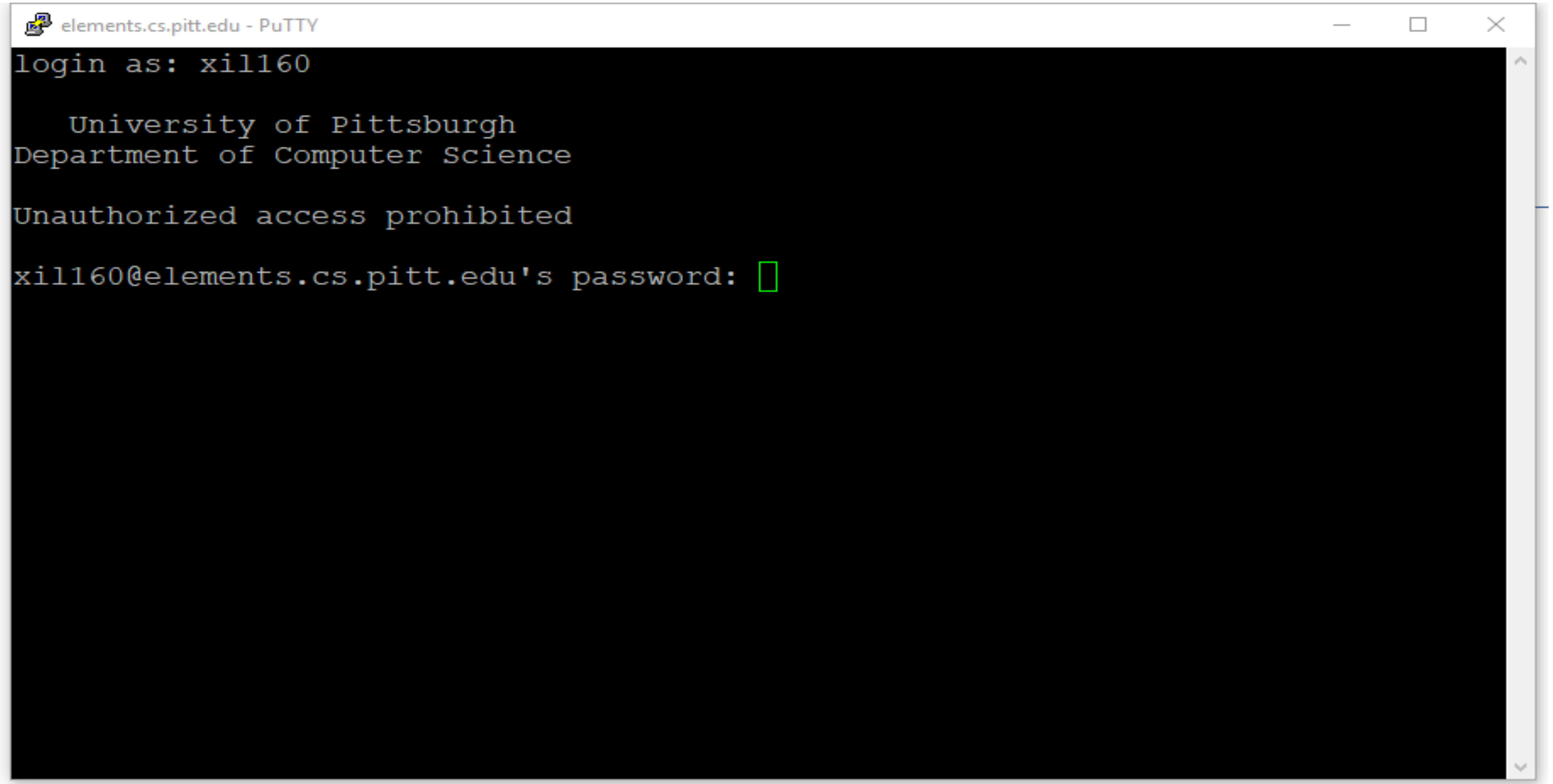
- Download from **www.putty.org**



```
login as: 
```

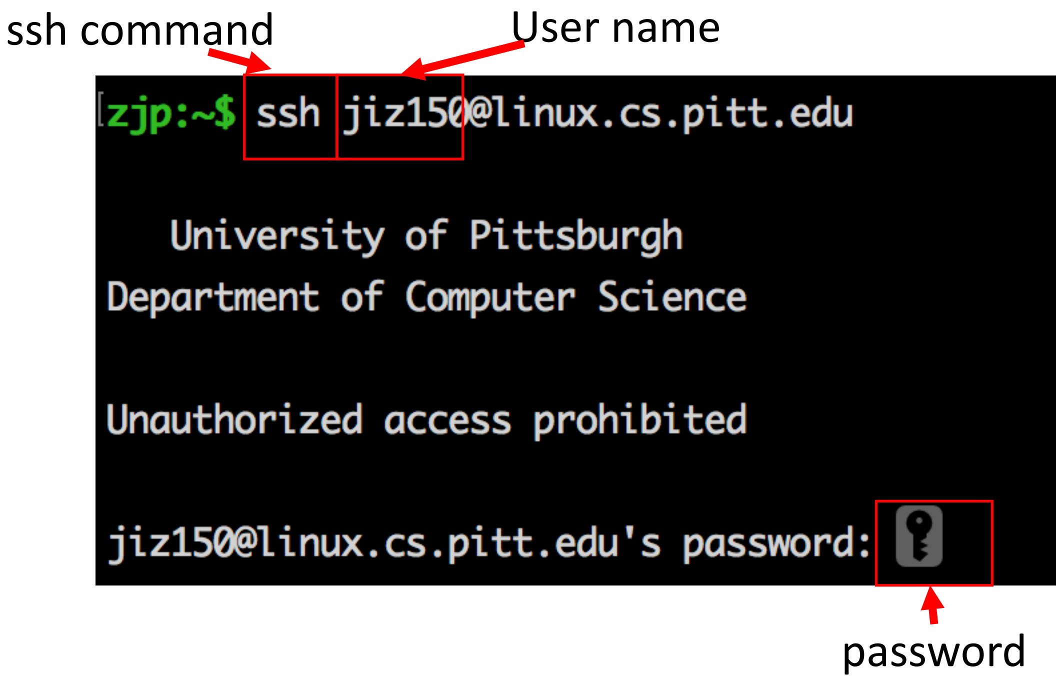
Windows - Putty

- Download from **www.putty.org**



```
login as: xil160

   University of Pittsburgh
Department of Computer Science

Unauthorized access prohibited

xil160@elements.cs.pitt.edu's password:
```

MacOS/Ubuntu - Terminal

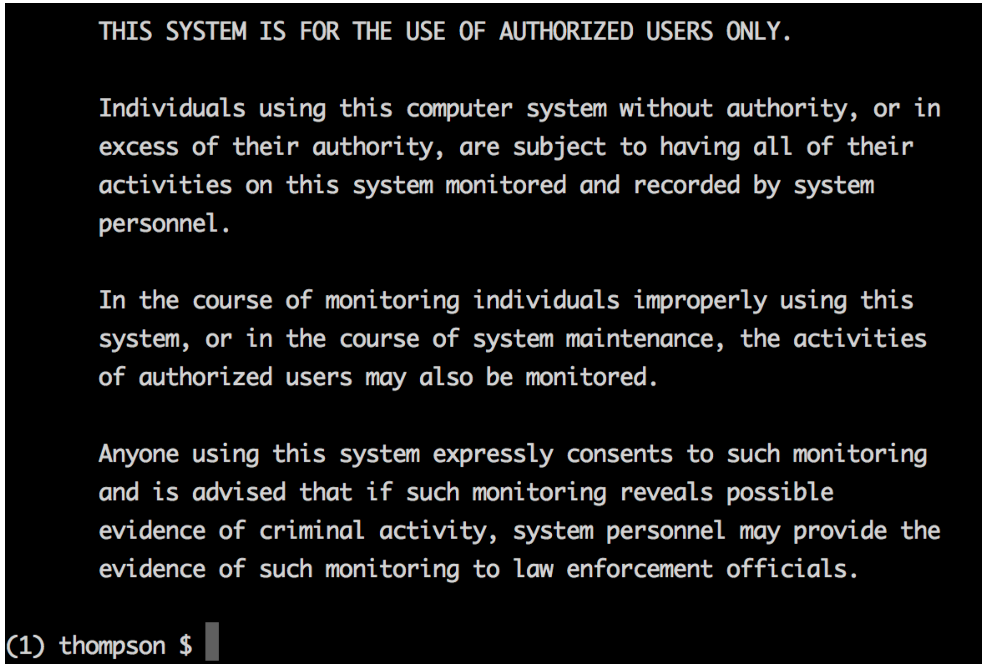


ssh command

User name

```
[zjp:~$ ssh jiz150@linux.cs.pitt.edu

   University of Pittsburgh
Department of Computer Science

Unauthorized access prohibited

jiz150@linux.cs.pitt.edu's password:
```

password



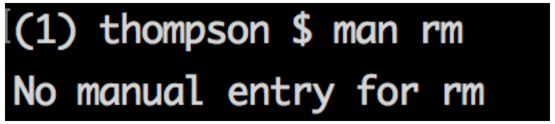
```
        THIS SYSTEM IS FOR THE USE OF AUTHORIZED USERS ONLY.

        Individuals using this computer system without authority, or in
        excess of their authority, are subject to having all of their
        activities on this system monitored and recorded by system
        personnel.

        In the course of monitoring individuals improperly using this
        system, or in the course of system maintenance, the activities
        of authorized users may also be monitored.

        Anyone using this system expressly consents to such monitoring
        and is advised that if such monitoring reveals possible
        evidence of criminal activity, system personnel may provide the
        evidence of such monitoring to law enforcement officials.

(1) thompson $
```

Basic Linux Shell commands

- Check Current Directory – **pwd**
- List directories - **ls**
- Create/Remove directory – **mkdir/rmdir**
- Remove files – **rm**
- Copy files from anywhere to anywhere – **cp**
  - cp **<current path> <new path>**
  - cp **some_text.txt Desktop/**
- Move files from anywhere to anywhere – **mv**
  - mv **<current path> <new path>**
  - mv **some_text.txt Desktop/**

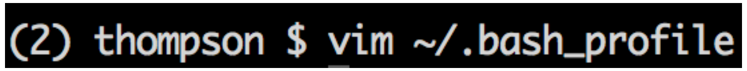
Basic Linux Shell commands

- Manual page
  - Type “man COMMAND” to see the manual page for a command
    - E.g., type “**man rm**” to see how to use “rm”. Type “q” to exit the man page
    - Possible problem:
      - IF: you saw “No manual entry” on linux.cs.pitt.edu

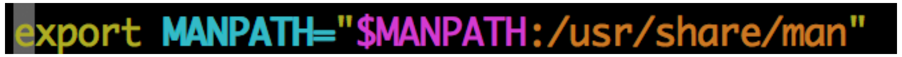


```
(1) thompson $ man rm
No manual entry for rm
```

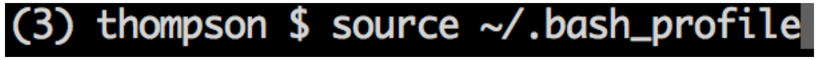
      - THEN: edit the “~/.bash_profile” file by adding the “export” line in the end of the file, then run “source ~/.bash_profile”



```
(2) thompson $ vim ~/.bash_profile
```



```
export MANPATH="$MANPATH:/usr/share/man"
```



```
(3) thompson $ source ~/.bash_profile
```

  - Or you can simply search something like “man page rm” online

Building projects

- Small programs
  - single file

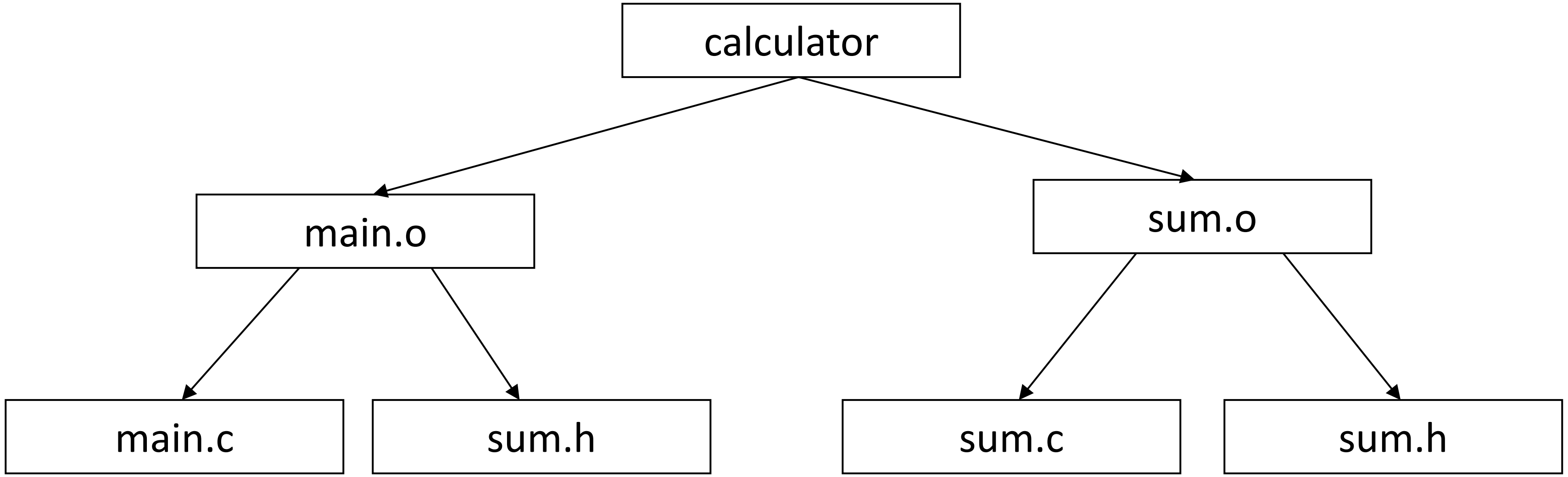
Building projects

- Small programs (easy to compile)
    - single file

```
~home-laptop->$ gcc main.c -o calculator
```

Building projects

- Small programs (easy to compile)
  - single file
- Bigger programs
  - multiple files

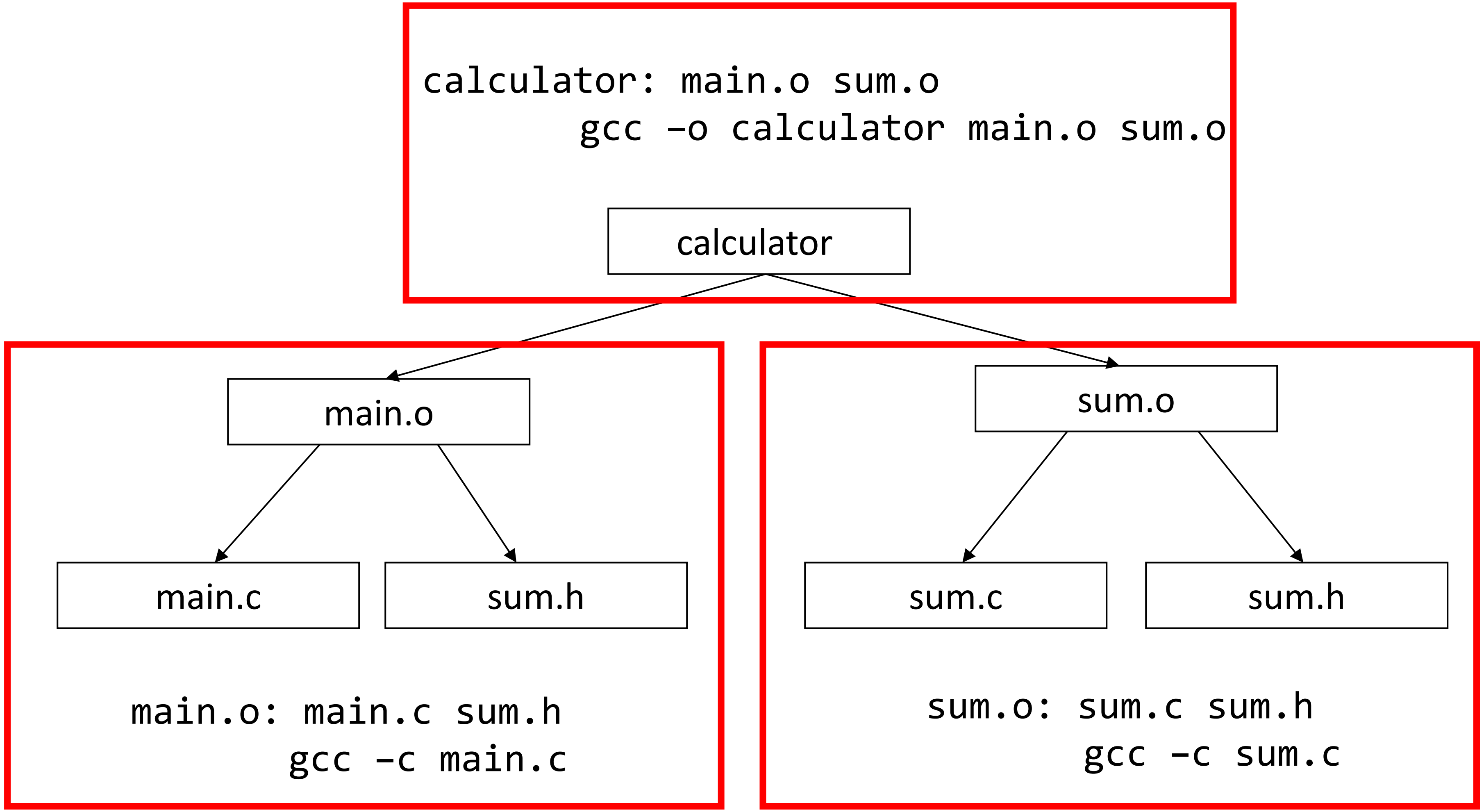


Multiple source files

Use the Makefile

```
TARGET: DEPENDENCIES
 ACTION
```

Use the Makefile



```
calculator: main.o sum.o
        gcc -o calculator main.o sum.o
```

```
main.o: main.c sum.h
        gcc -c main.c
```

```
sum.o: sum.c sum.h
        gcc -c sum.c
```

Makefile example

```
calculator: main.o sum.o
 gcc –o calculator main.o sum.o

main.o: main.c sum.h
 gcc –c main.c

sum.o: sum.c sum.h
 gcc –c sum.c
```

The Makefile

- Running “make”

```
~home-laptop->$ make calculator
```

The Makefile

```
calculator: main.o sum.o
 gcc –o calculator main.o sum.o
main.o: main.c sum.h
 gcc –c main.c
sum.o: sum.c sum.h
 gcc –c sum.c
clean:
    rm -f *.o
```

The Makefile

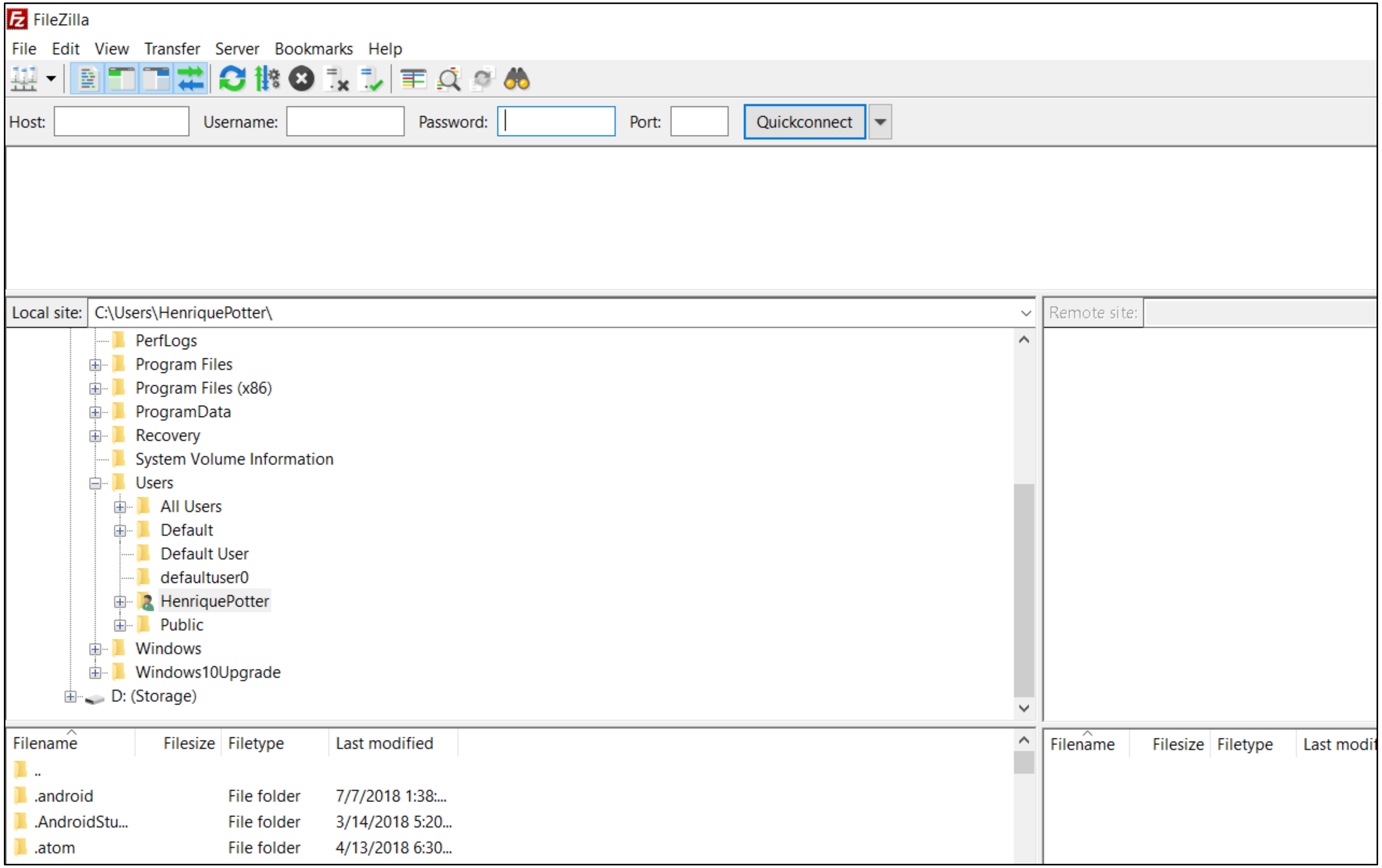
- Running “make”

```
~home-laptop->$ make clean
```

Upload to / download from server

- FileZilla – Windows/MacOS
  - GUI based
- Or use the “scp” command
  - Try check scp’s man page

GUI based FTP Clients



Host: Username: Password: Port:

Local site: C:\Users\HenriquePotter\

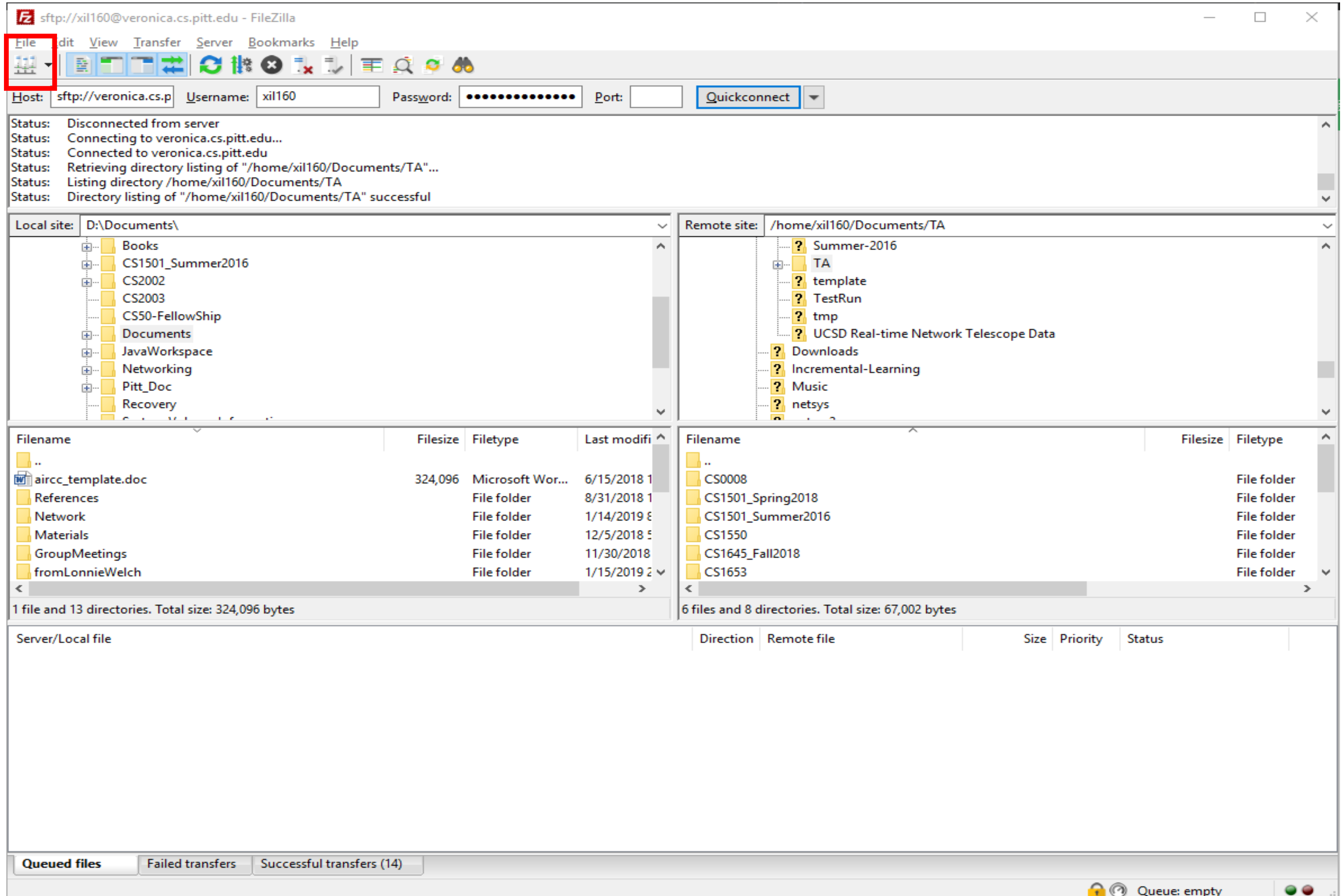
Remote site:

```
PerfLogs
Program Files
Program Files (x86)
ProgramData
Recovery
System Volume Information
Users
    All Users
    Default
    Default User
    defaultuser0
    HenriquePotter
    Public
Windows
Windows10Upgrade
D: (Storage)
```

Filename	Filesize	Filetype	Last modified
..			
.android		File folder	7/7/2018 1:38:...
.AndroidStu...		File folder	3/14/2018 5:20...
.atom		File folder	4/13/2018 6:30...

Filename	Filesize	Filetype	Last modi

GUI based FTP Clients



sftp://xil160@veronica.cs.pitt.edu - FileZilla

Host: sftp://veronica.cs.p Username: xil160 Password: ••••••••••••••• Port:

```
Status:    Disconnected from server
Status:    Connecting to veronica.cs.pitt.edu...
Status:    Connected to veronica.cs.pitt.edu
Status:    Retrieving directory listing of "/home/xil160/Documents/TA"...
Status:    Listing directory /home/xil160/Documents/TA
Status:    Directory listing of "/home/xil160/Documents/TA" successful
```

Local site: D:\Documents\

- Books
- CS1501_Summer2016
- CS2002
- CS2003
- CS50-FellowShip
- Documents
- JavaWorkspace
- Networking
- Pitt_Doc
- Recovery

Remote site: /home/xil160/Documents/TA

- Summer-2016
- TA
- template
- TestRun
- tmp
- UCSD Real-time Network Telescope Data
- Downloads
- Incremental-Learning
- Music
- netsys

Filename	Filesize	Filetype	Last modifi
..			
aircc_template.doc	324,096	Microsoft Wor...	6/15/2018 1
References		File folder	8/31/2018 1
Network		File folder	1/14/2019 8
Materials		File folder	12/5/2018 5
GroupMeetings		File folder	11/30/2018
fromLonnieWelch		File folder	1/15/2019 2

1 file and 13 directories. Total size: 324,096 bytes

Filename	Filesize	Filetype
..		
CS0008		File folder
CS1501_Spring2018		File folder
CS1501_Summer2016		File folder
CS1550		File folder
CS1645_Fall2018		File folder
CS1653		File folder

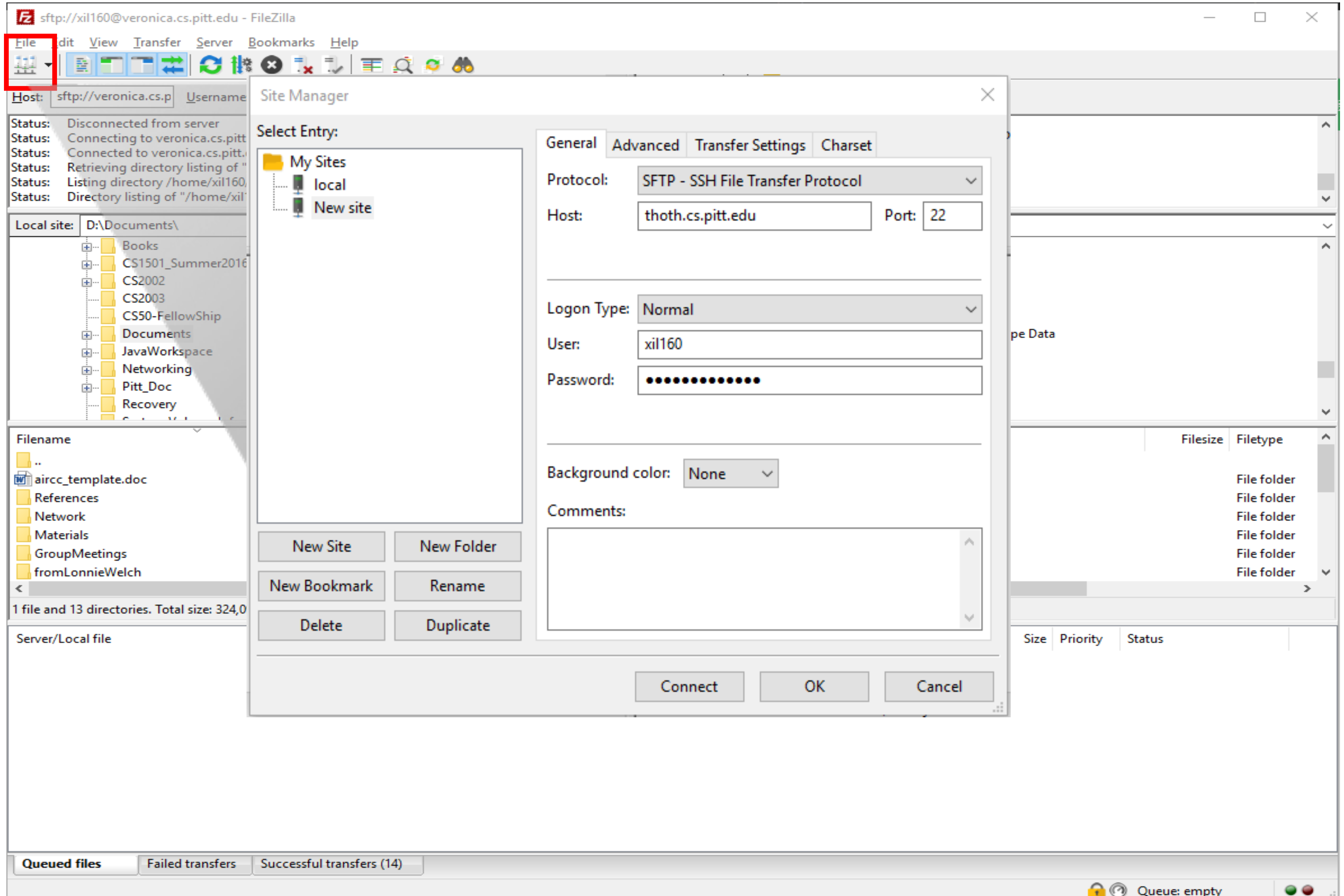
6 files and 8 directories. Total size: 67,002 bytes

Server/Local file	Direction	Remote file	Size	Priority	Status

Queued files | Failed transfers | Successful transfers (14)

Queue: empty

GUI based FTP Clients



```
Status:   Disconnected from server
Status:   Connecting to veronica.cs.pitt
Status:   Connected to veronica.cs.pitt.
Status:   Retrieving directory listing of "
Status:   Listing directory /home/xil160,
Status:   Directory listing of "/home/xil
```

Local site: D:\Documents\

- Books
- CS1501_Summer2016
- CS2002
- CS2003
- CS50-FellowShip
- Documents
- JavaWorkspace
- Networking
- Pitt_Doc
- Recovery

Filename
..
aircc_template.doc
References
Network
Materials
GroupMeetings
fromLonnieWelch

1 file and 13 directories. Total size: 324,0

Filesize	Filetype
	File folder
	File folder
	File folder
	File folder
	File folder
	File folder

Server/Local file	Size	Priority	Status

pe Data

Protocol: SFTP - SSH File Transfer Protocol

Host: thoth.cs.pitt.edu Port: 22

Logon Type: Normal

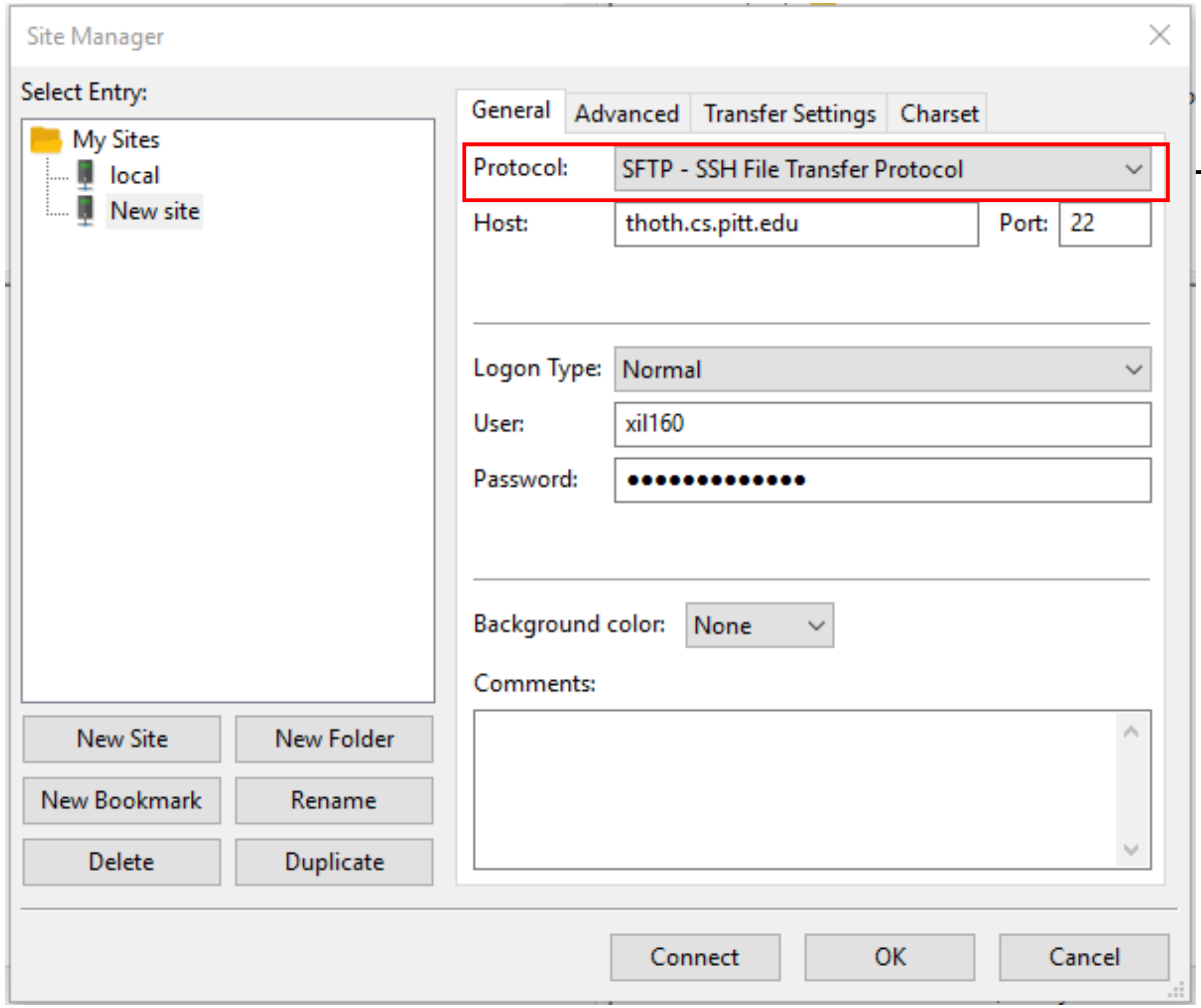
User: xil160

Password: ••••••••••••••

Background color: None

Comments:

GUI based FTP Clients



Protocol: SFTP - SSH File Transfer Protocol

Host: thoth.cs.pitt.edu Port: 22

Logon Type: Normal

User: xil160

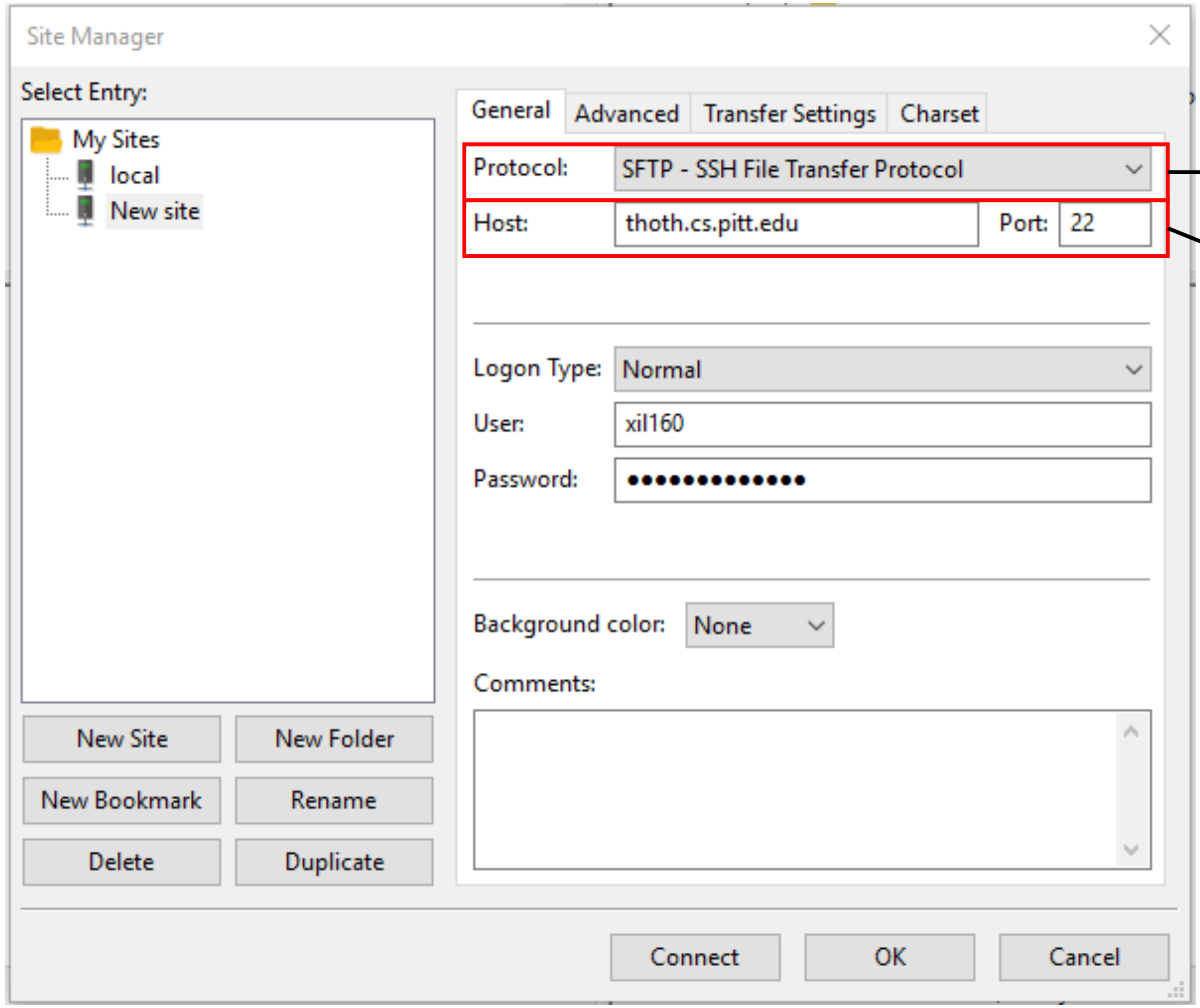
Password: ••••••••••••••

Background color: None

Comments:

Select **SFTP**

GUI based FTP Clients



Site Manager

Select Entry:

Protocol: SFTP - SSH File Transfer Protocol

Host: thoth.cs.pitt.edu Port: 22

Logon Type: Normal

User: xil160

Password: ●●●●●●●●●●●●●●

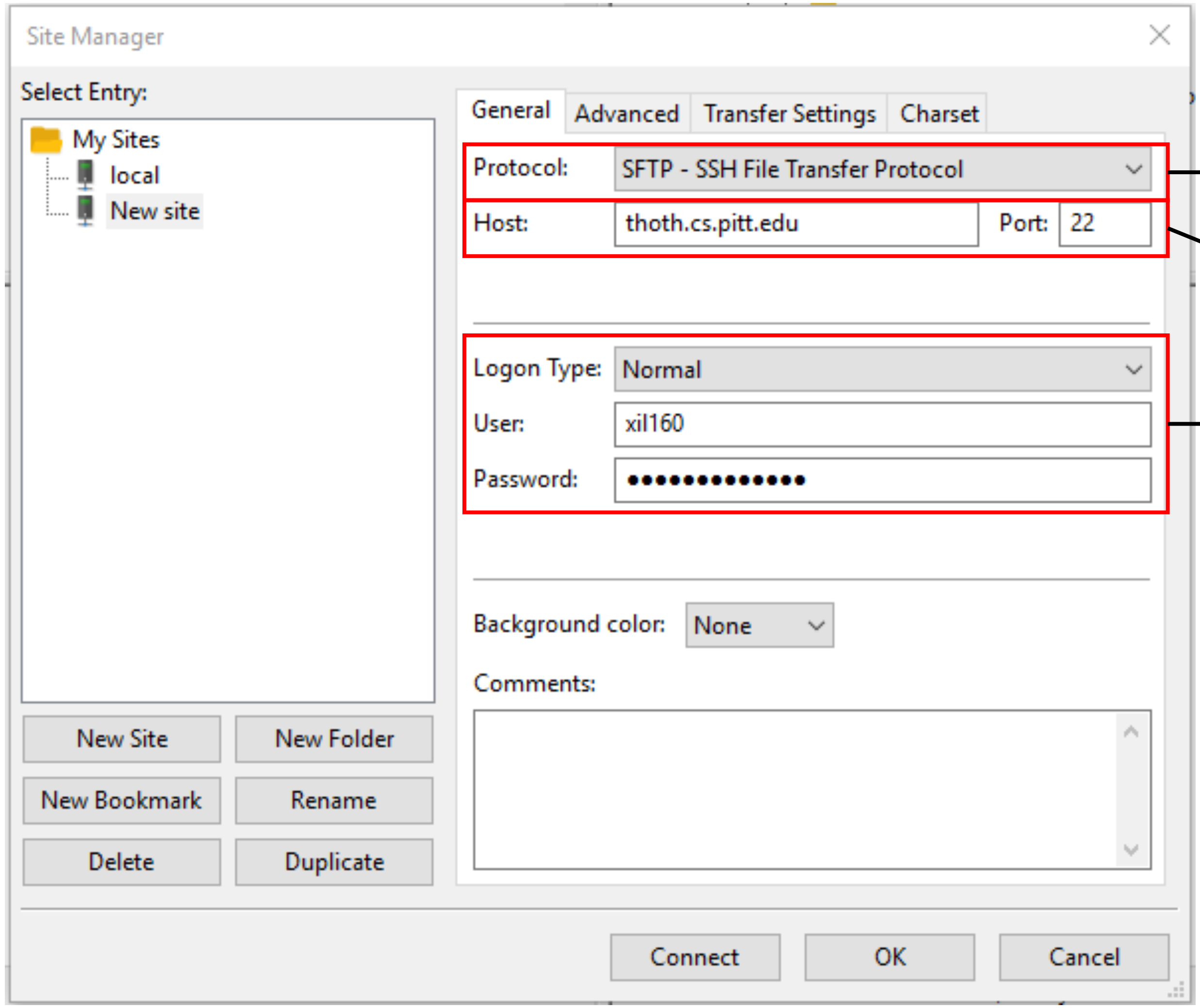
Background color: None

Comments:

Select **SFTP**

Server info

GUI based FTP Clients



Protocol: SFTP - SSH File Transfer Protocol

Host: thoth.cs.pitt.edu Port: 22

Logon Type: Normal

User: xil160

Password: ••••••••••••••

Background color: None

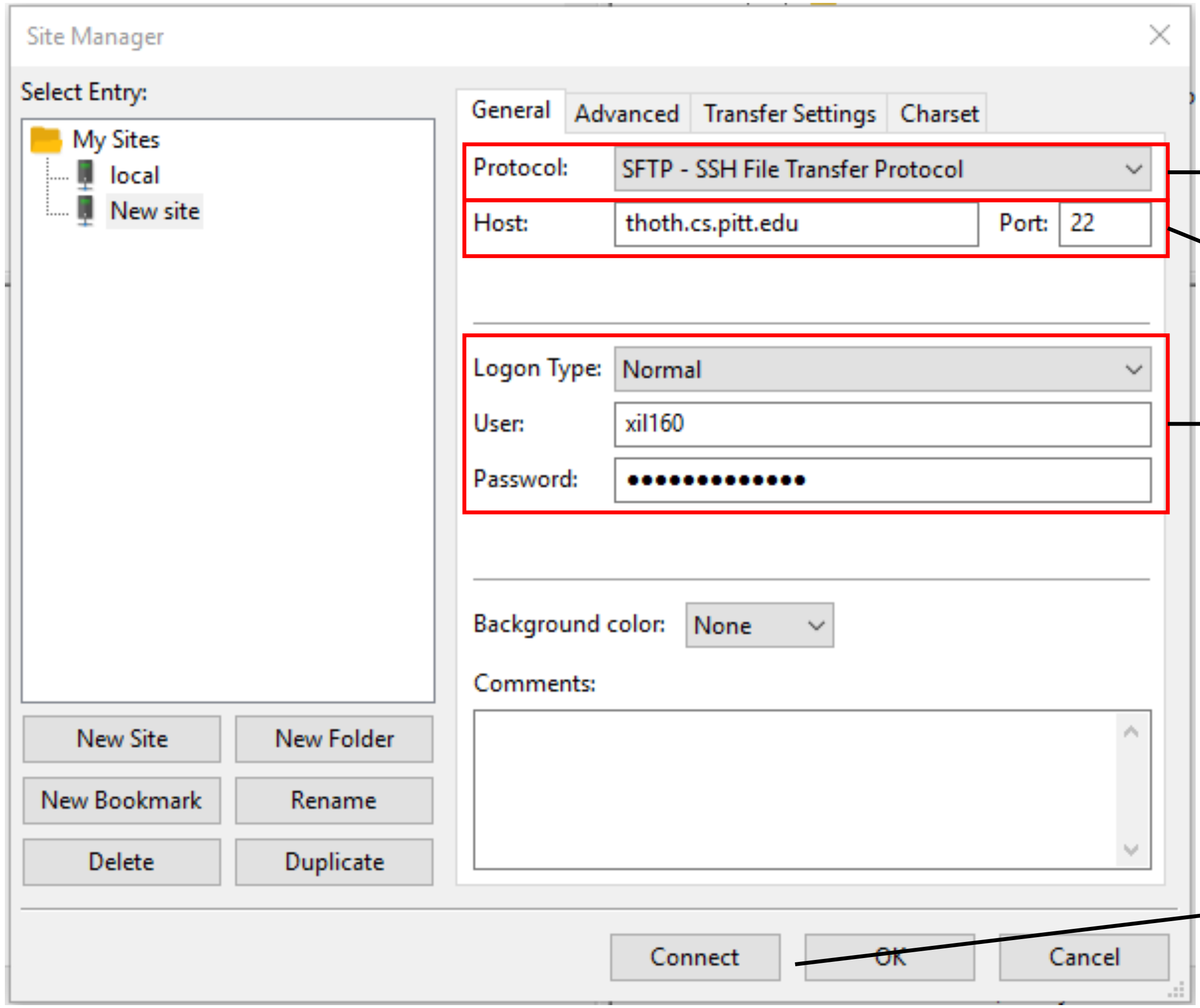
Comments:

Select **SFTP**

Server info

normal Logon

GUI based FTP Clients



Protocol: SFTP - SSH File Transfer Protocol

Host: thoth.cs.pitt.edu   Port: 22

Logon Type: Normal

User: xil160

Password: ••••••••••••••

Background color: None

Comments:

Select **SFTP**

Server info

normal Logon

Select **Connect**

Using Atom

- Good coding tool
  - Easily search for variables names in any file inside a folder
  - Lots of plugins
- Synchronize Files
  - Install ***remote-sync*** (recommended)
  - Right click main project folder
  - Navigate to Remote Sync > Configure
  - Fill in the details / select options
  - Hit save
  - https://atom.io/packages/remote-sync

C Pointers

- Pointers are variables that contain *memory addresses* as their values.
- A variable name *directly* reference a value.
- A pointer *indirectly* reference a value.
- A pointer variable must be declared before it can be used.

C Pointers

- Examples of pointer declarations:
  - int *a;
  - float *b;

* tells the compiler that the variable is to be a pointer, and the type of data that the pointer points to

- & and *:
  - & -- “address operator” gives or produces the memory address of a data variable.
  - * -- “dereferencing operator” provides the contents in the memory location specified by a pointer.

C Pointers

- A simple Example

```
int number;
int *ptr1;
int *ptr2;

ptr1 = &number;
number = 2;
ptr2 = &number;
printf("\n*ptr1 = %d\t*ptr2 = %d\n", *ptr1, *ptr2);
```

C Pointers

- Arrays

```
int demoArray[5] = {8, 19, 34, 0, 3};
printf("Element \t Address \t Value \n");

for (int i = 0; i < 5; i++) {
    printf("demoArray[%d]\t%p\t%d\n", i, &demoArray[i], demoArray[i]);
}

//array names are just pointers to the first Element
printf("\ndemoArray\t\t%p\n", demoArray);

//dereference
printf("\n*demoArray\t\t%d\n", *demoArray);
printf("\n*(demoArray+2)\t\t%d\n", *(demoArray+2));
```

System Call

-

exercise

CS 1550 – xv6

- Simple Unix-like **operating system** for teaching, developed in 2006.
  - Provides basic services for running programs
  - https://pdos.csail.mit.edu/6.828/2012/xv6.html?utm_source=dlvr.it&utm_medium=tumblr

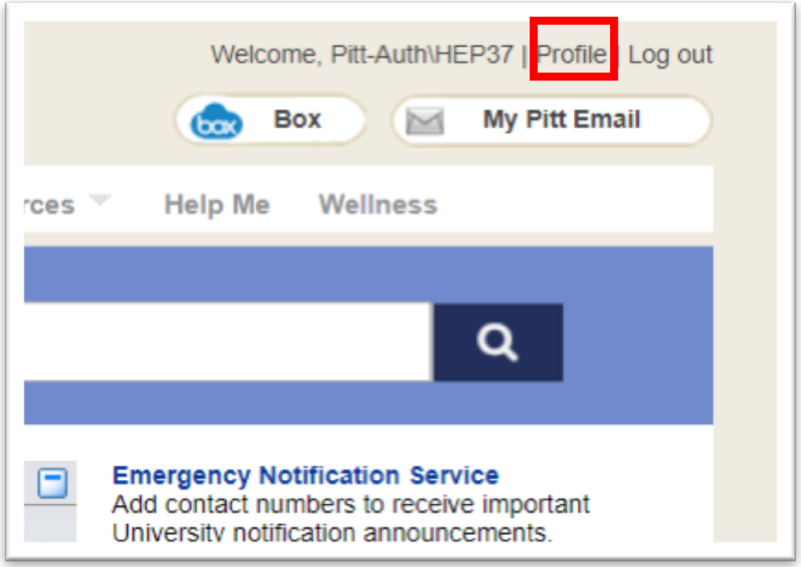
xv6 OS

CS 1550 – xv6

- Has a **subset of traditional** system calls
  - **fork**() Create process
  - **exit**() Terminate current process
  - **wait**() Wait for a child process
  - **kill**(pid) Terminate process pid
  - **getpid**() Return current process's id
  - **sleep(n)** Sleep for n time units
  - **exec(filename, *argv)** Load a file and execute it sbrk(n)
  - ….

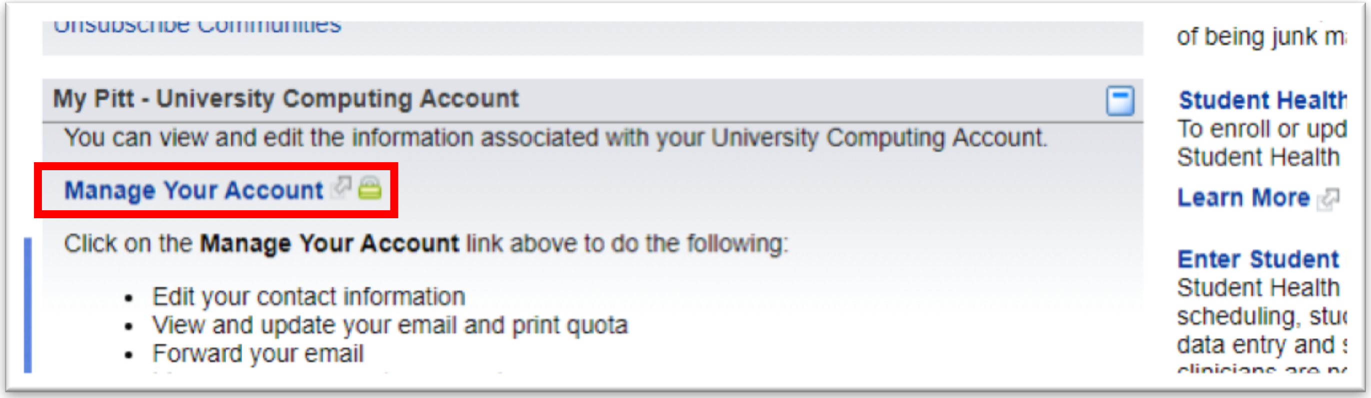
CS 1550 – Compile and Run xv6

1. Extend disk Quota, if you have less then 500mb free space
   a) Log in to https://my.pitt.edu
   b) Click on "Profile" at the top-right corner of the screen
   c) Click on "Manage Your Account"
   d) Click on "EMAIL & MESSAGING" -> "UNIX QUOTA“
   e) Click on "Increase My UNIX Quota"



Emergency Notification Service

Add contact numbers to receive important University notification announcements.



My Pitt - University Computing Account

You can view and edit the information associated with your University Computing Account.

Manage Your Account

Click on the **Manage Your Account** link above to do the following:

- Edit your contact information
- View and update your email and print quota
- Forward your email

of being junk m

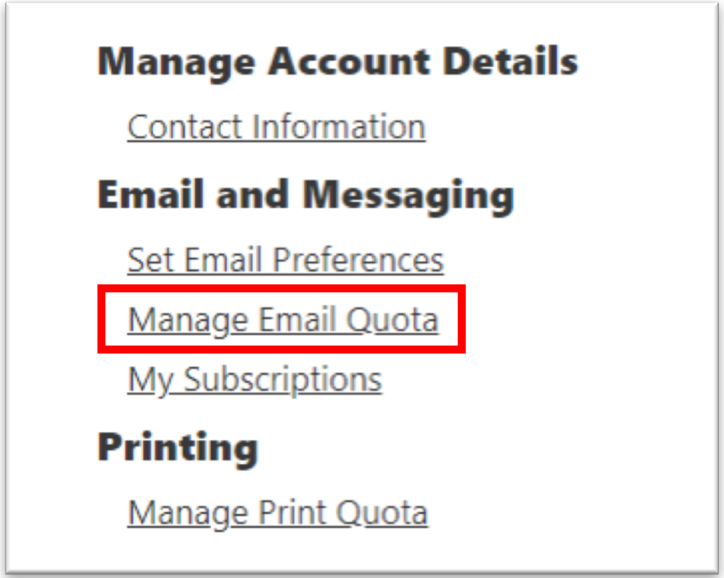
Student Health

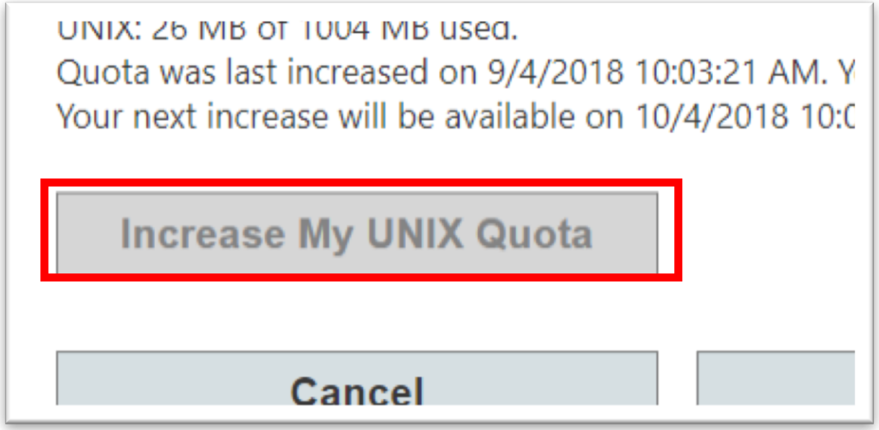
To enroll or upd Student Health

Student Health scheduling, stu data entry and

CS 1550 – Compile and Run xv6

1. Extend disk Quota, if you have less then 500mb free space
   a) Log in to https://my.pitt.edu
   b) Click on "Profile" at the top-right corner of the screen
   c) Click on "Manage Your Account"
   d) Click on "EMAIL & MESSAGING" -> "UNIX QUOTA“
   e) Click on "Increase My UNIX Quota"

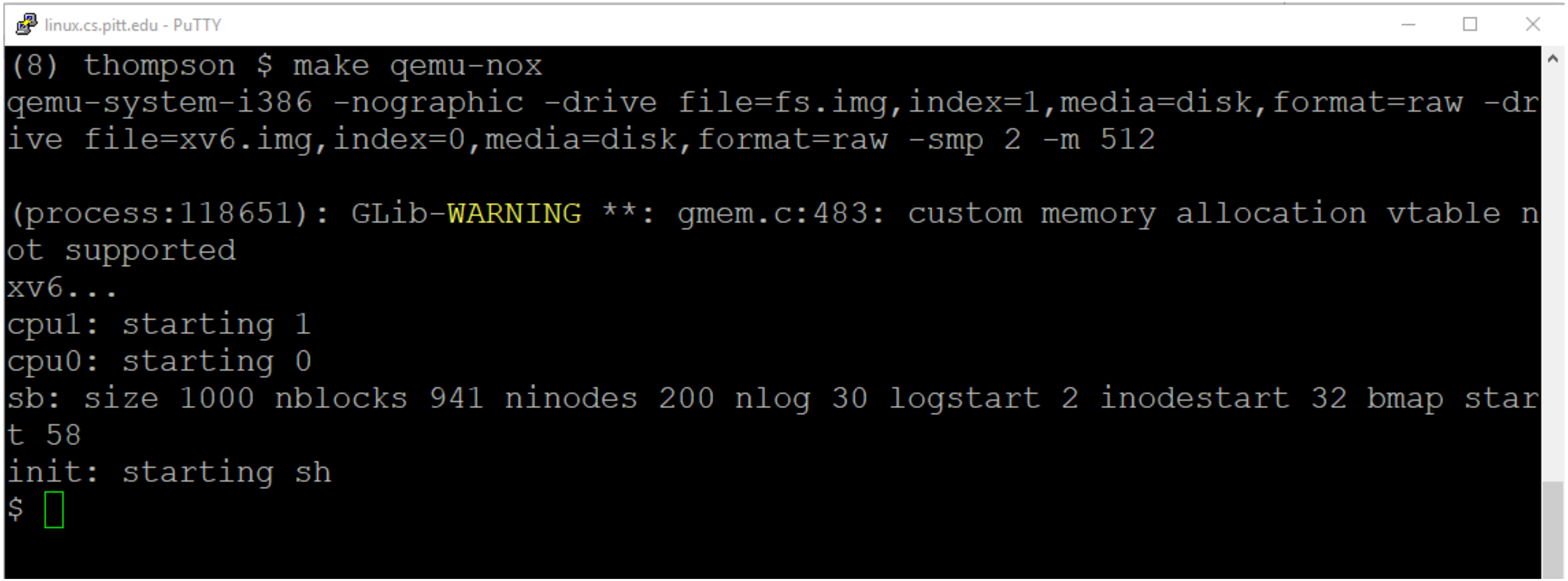



UNIX: 26 MB of 1004 MB used.
Quota was last increased on 9/4/2018 10:03:21 AM. Y
Your next increase will be available on 10/4/2018 10:(

CS 1550 – xv6

- Log in to **linux.cs.pitt.edu**
  - **ssh** user_name**@linux.cs.pitt.edu**
- Download the xv6 source code from github
  - **git clone git://github.com/mit-pdos/xv6-public.git**
- Get into the cloned xv6 source code folder
  - **cd xv6-public**
- Compile and run the code with
  - **make qemu-nox** → **Compiles** and **run xv6** with **qemu**
  - qemu-nox run the console version of the emulator

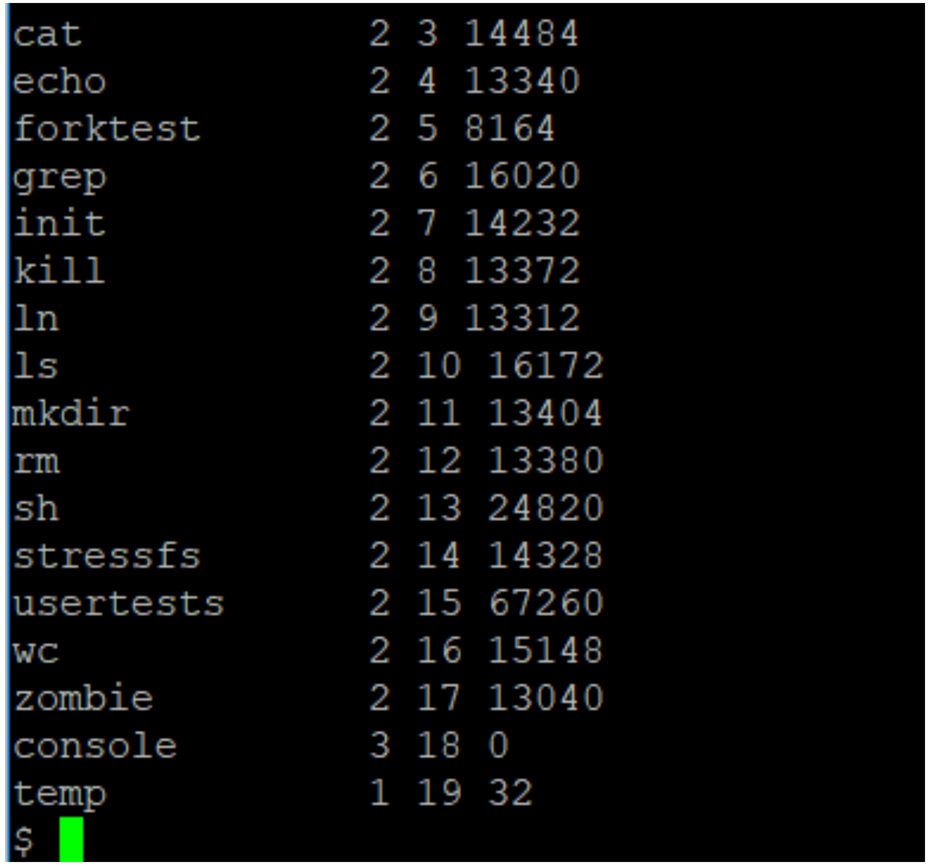
CS 1550 – xv6



```
(8) thompson $ make qemu-nox
qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -dr
ive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512

(process:118651): GLib-WARNING **: gmem.c:483: custom memory allocation vtable n
ot supported
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
$ 
```

CS 1550 – xv6

- Once in xv6 you can run **ls**



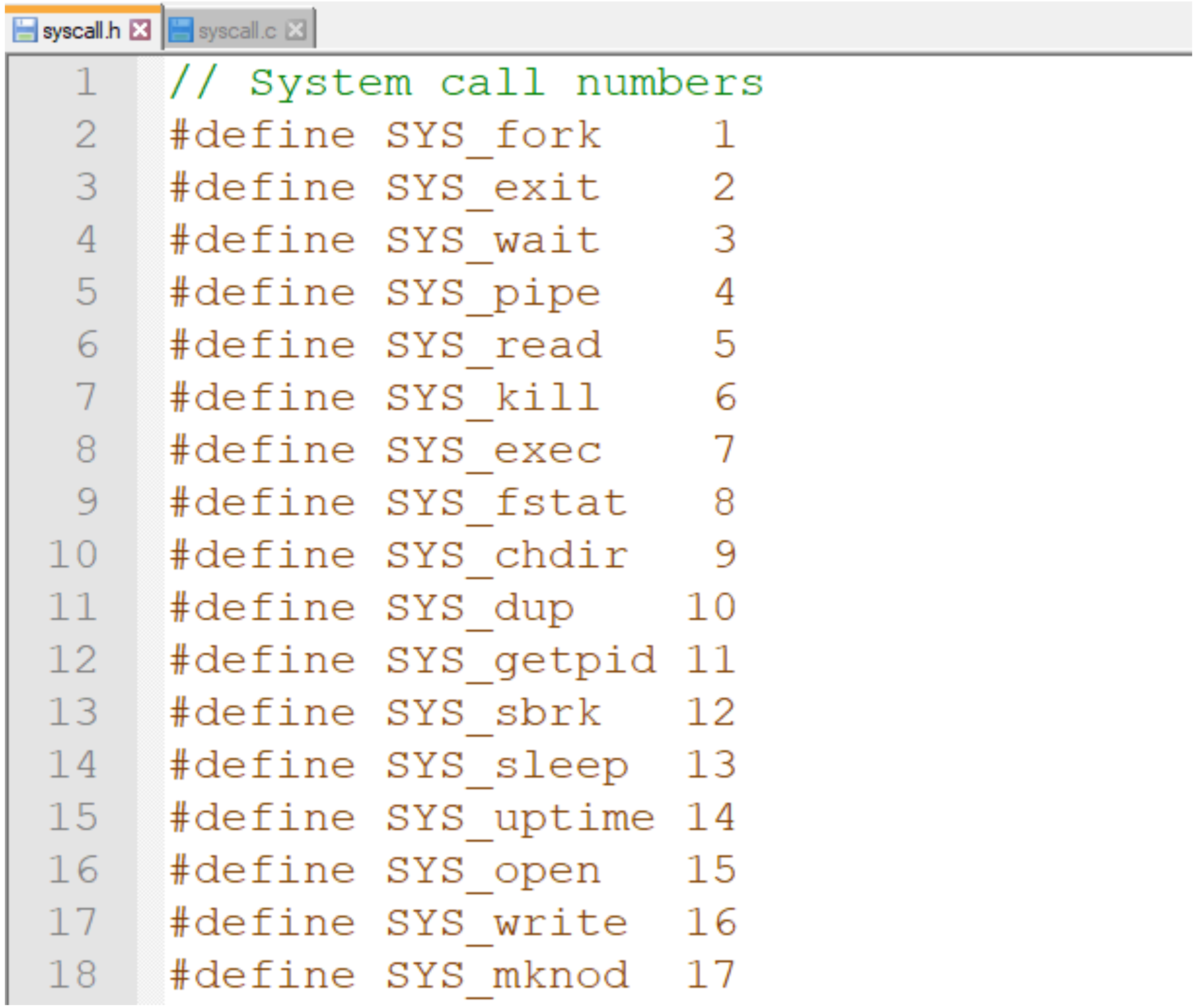
```
cat            2 3 14484
echo           2 4 13340
forktest       2 5 8164
grep           2 6 16020
init           2 7 14232
kill           2 8 13372
ln             2 9 13312
ls             2 10 16172
mkdir          2 11 13404
rm             2 12 13380
sh             2 13 24820
stressfs       2 14 14328
usertests      2 15 67260
wc             2 16 15148
zombie         2 17 13040
console        3 18 0
temp           1 19 32
$ 
```

CS 1550 – xv6 – Adding a custom Syscall

- In Lab1, you need to add a syscall “getcount”.
- As an example, let’s try to add a syscall named “getday”, which returns a hardcoded pseudo date

CS 1550 – xv6 – Adding a custom Syscall

- First, we need to define our new call and its number at
  - **syscall.h**

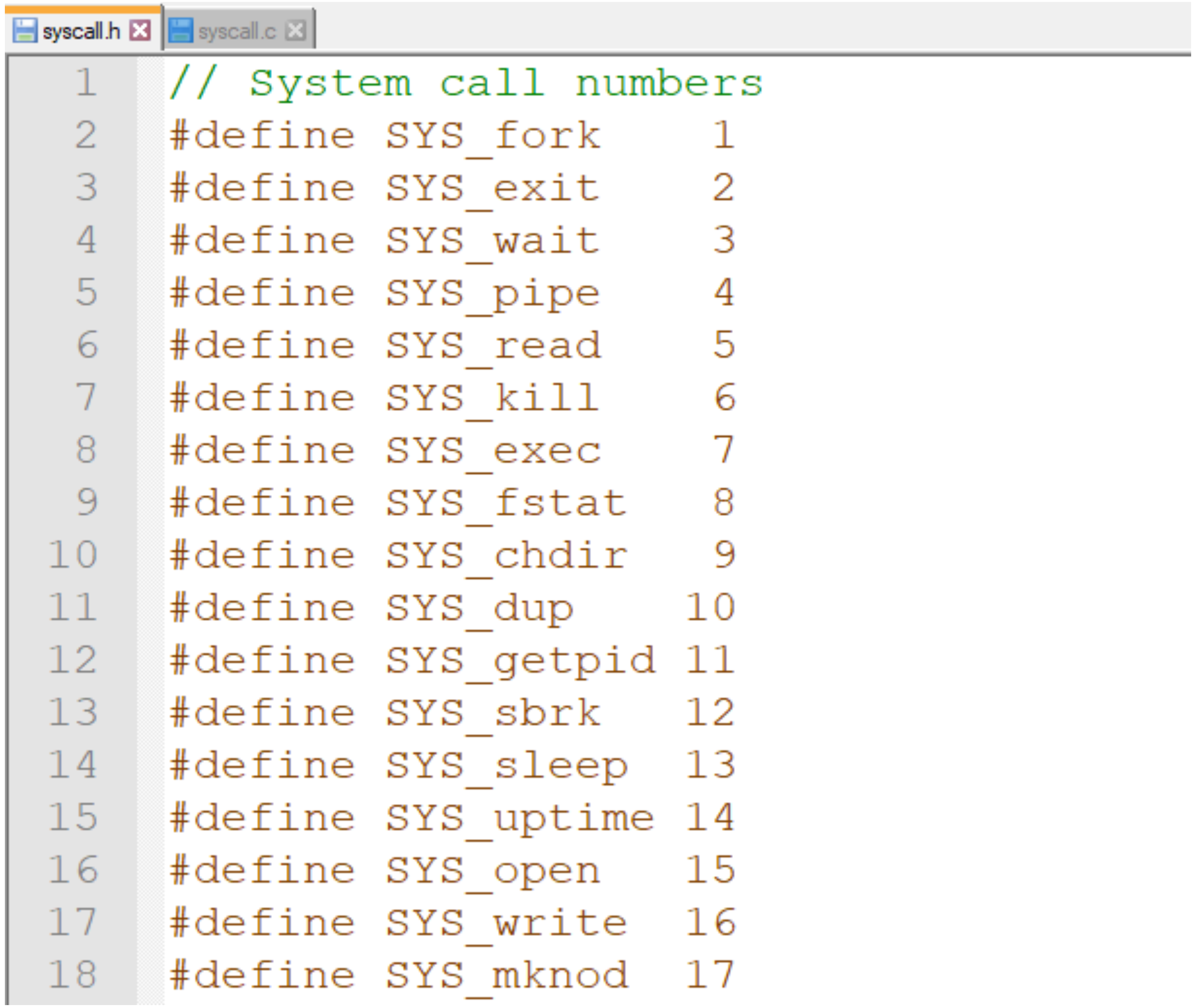


```
// System call numbers
#define SYS_fork    1
#define SYS_exit    2
#define SYS_wait    3
#define SYS_pipe    4
#define SYS_read    5
#define SYS_kill    6
#define SYS_exec    7
#define SYS_fstat   8
#define SYS_chdir   9
#define SYS_dup    10
#define SYS_getpid 11
#define SYS_sbrk   12
#define SYS_sleep  13
#define SYS_uptime 14
#define SYS_open   15
#define SYS_write  16
#define SYS_mknod  17
```

CS 1550 – xv6 – Adding a custom Syscall

- First, we need to define our new call and its number at
  - **syscall.h**

- Add
  - #define SYS_getday 22

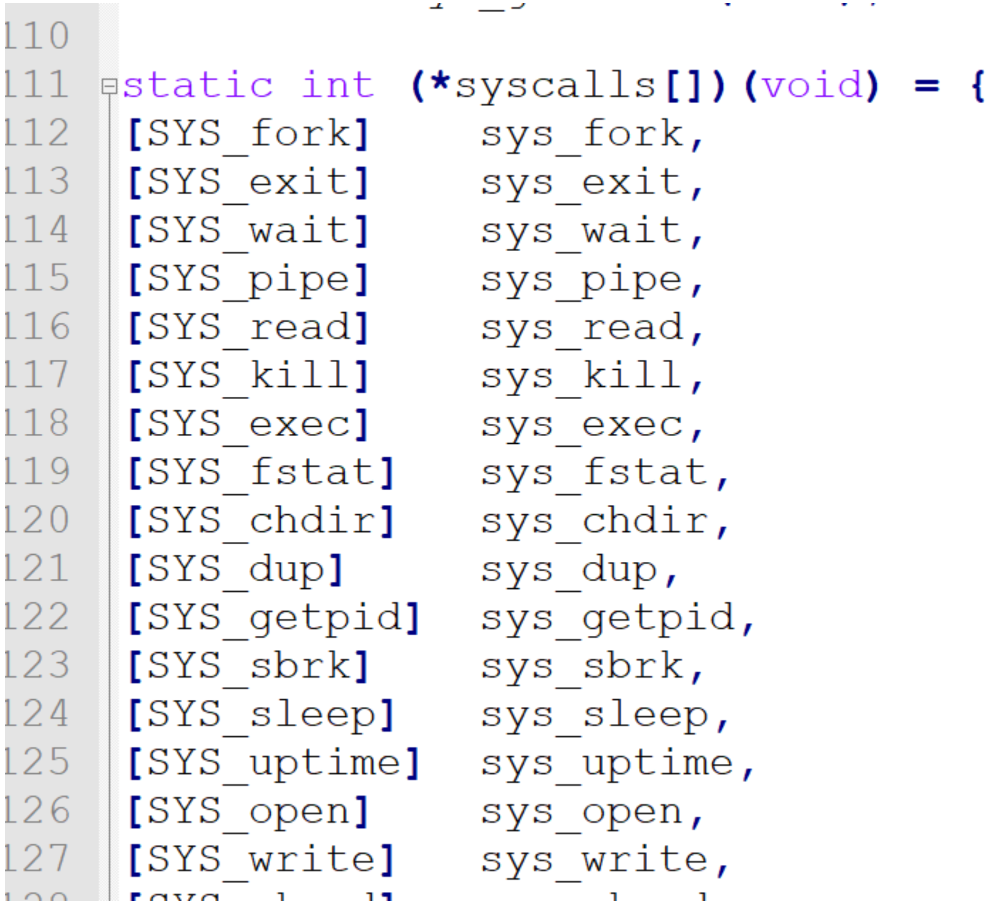


```
// System call numbers
#define SYS_fork    1
#define SYS_exit    2
#define SYS_wait    3
#define SYS_pipe    4
#define SYS_read    5
#define SYS_kill    6
#define SYS_exec    7
#define SYS_fstat   8
#define SYS_chdir   9
#define SYS_dup    10
#define SYS_getpid 11
#define SYS_sbrk   12
#define SYS_sleep  13
#define SYS_uptime 14
#define SYS_open   15
#define SYS_write  16
#define SYS_mknod  17
```

CS 1550 – xv6 – Adding a custom Syscall

- Next, we need to map the new call in the array pointer of system calls
  - **syscall.c**

- Add
  - [SYS_getday]   sys_getday,



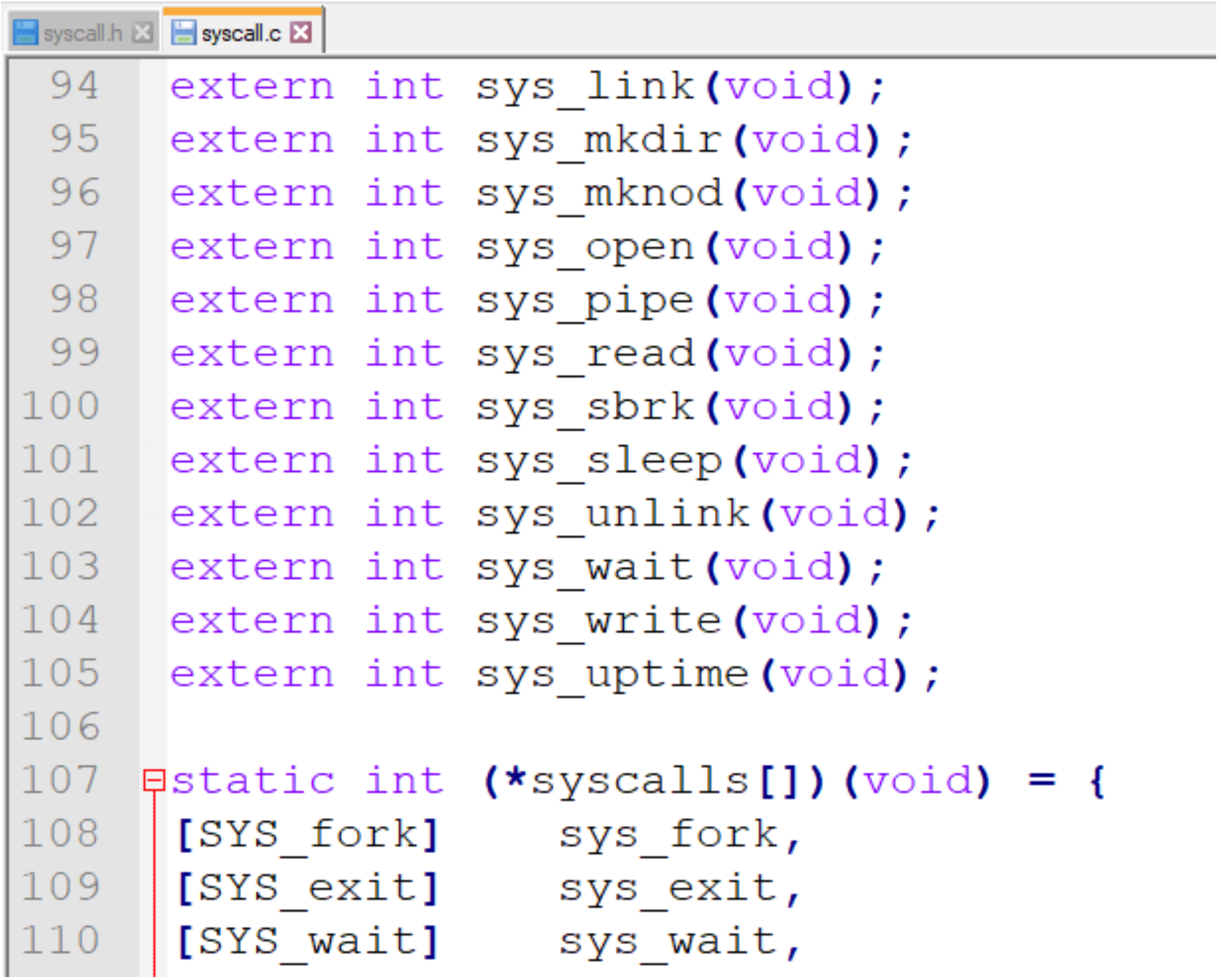
```
static int (*syscalls[])(void) = {
[SYS_fork]    sys_fork,
[SYS_exit]    sys_exit,
[SYS_wait]    sys_wait,
[SYS_pipe]    sys_pipe,
[SYS_read]    sys_read,
[SYS_kill]    sys_kill,
[SYS_exec]    sys_exec,
[SYS_fstat]   sys_fstat,
[SYS_chdir]   sys_chdir,
[SYS_dup]     sys_dup,
[SYS_getpid]  sys_getpid,
[SYS_sbrk]    sys_sbrk,
[SYS_sleep]   sys_sleep,
[SYS_uptime]  sys_uptime,
[SYS_open]    sys_open,
[SYS_write]   sys_write,
```

CS 1550 – xv6 – Adding a custom Syscall

- Next, we need to map the new call in the array pointer of system calls
  - **syscall.c**

- Add
  - [SYS_getday]   sys_getday,

- Add
  - extern int sys_getday(void);



```
extern int sys_link(void);
extern int sys_mkdir(void);
extern int sys_mknod(void);
extern int sys_open(void);
extern int sys_pipe(void);
extern int sys_read(void);
extern int sys_sbrk(void);
extern int sys_sleep(void);
extern int sys_unlink(void);
extern int sys_wait(void);
extern int sys_write(void);
extern int sys_uptime(void);

static int (*syscalls[])(void) = {
[SYS_fork]    sys_fork,
[SYS_exit]    sys_exit,
[SYS_wait]    sys_wait,
```

CS 1550 – xv6 – Adding a custom Syscall

- Then we need to implement the actual method

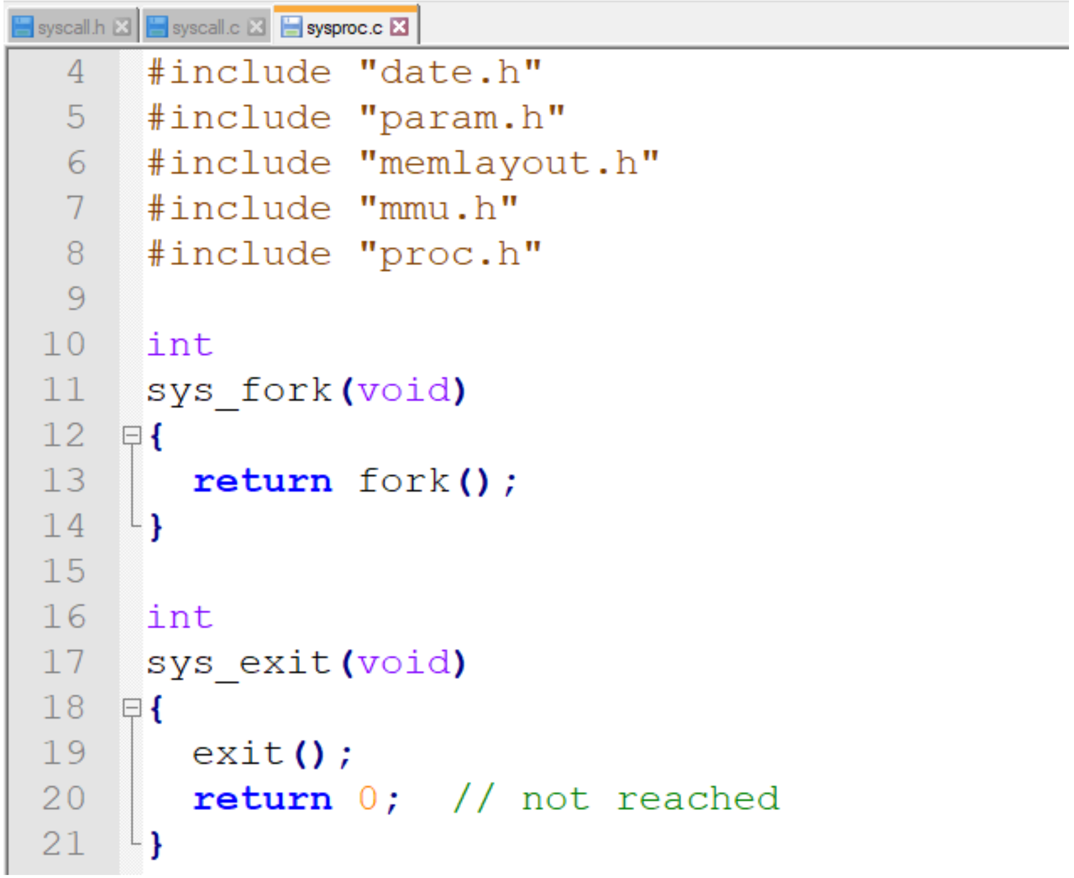
- In xv6 this is organized in two files.
  - sysfile.c -> file related system calls
  - **sysproc.c -> all the other syscalls**

CS 1550 – xv6 – Adding a custom Syscall

- Then we need to implement the actual method

- In xv6 this is organized in two files.
  - sysfile.c -> file related system calls
  - **sysproc.c -> all the other syscalls**

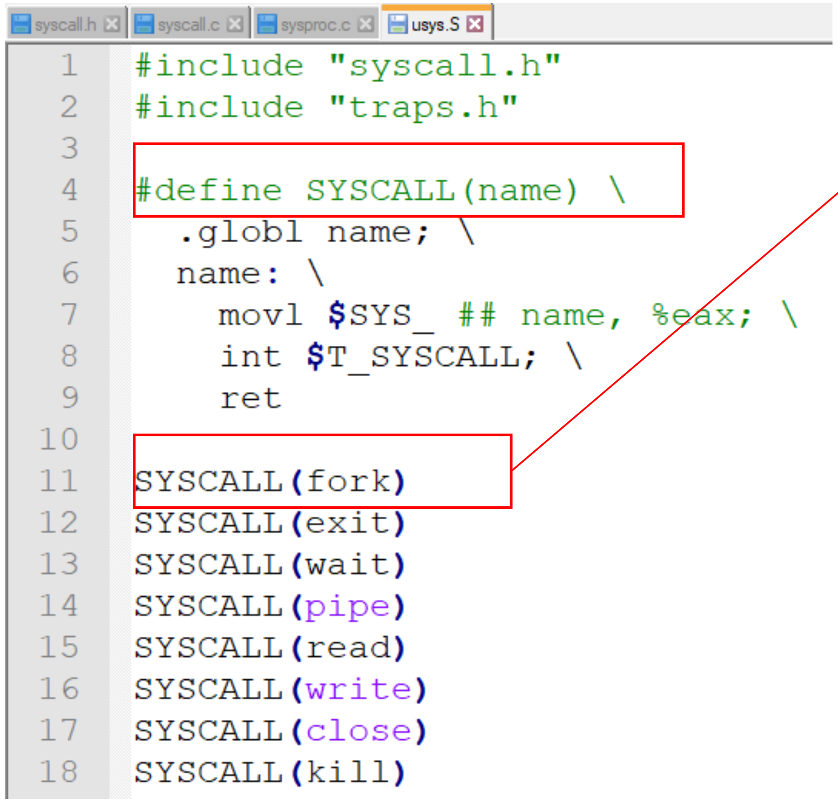
```
int
sys_getday(void)
{
   return 6;
}
```



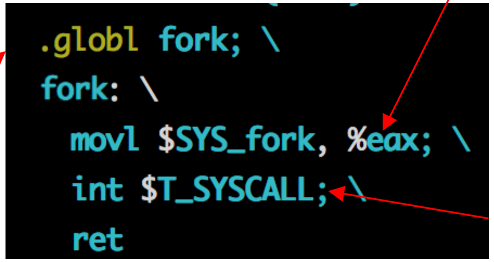
```
#include "date.h"
#include "param.h"
#include "memlayout.h"
#include "mmu.h"
#include "proc.h"

int
sys_fork(void)
{
  return fork();
}

int
sys_exit(void)
{
  exit();
  return 0;  // not reached
}
```

CS 1550 – xv6 – Adding a custom Syscall

- Afterwards we define the interface for user programs to call
  - Open usys.S



```
#include "syscall.h"
#include "traps.h"

#define SYSCALL(name) \
  .globl name; \
  name: \
    movl $SYS_ ## name, %eax; \
    int $T_SYSCALL; \
    ret

SYSCALL(fork)
SYSCALL(exit)
SYSCALL(wait)
SYSCALL(pipe)
SYSCALL(read)
SYSCALL(write)
SYSCALL(close)
SYSCALL(kill)
```



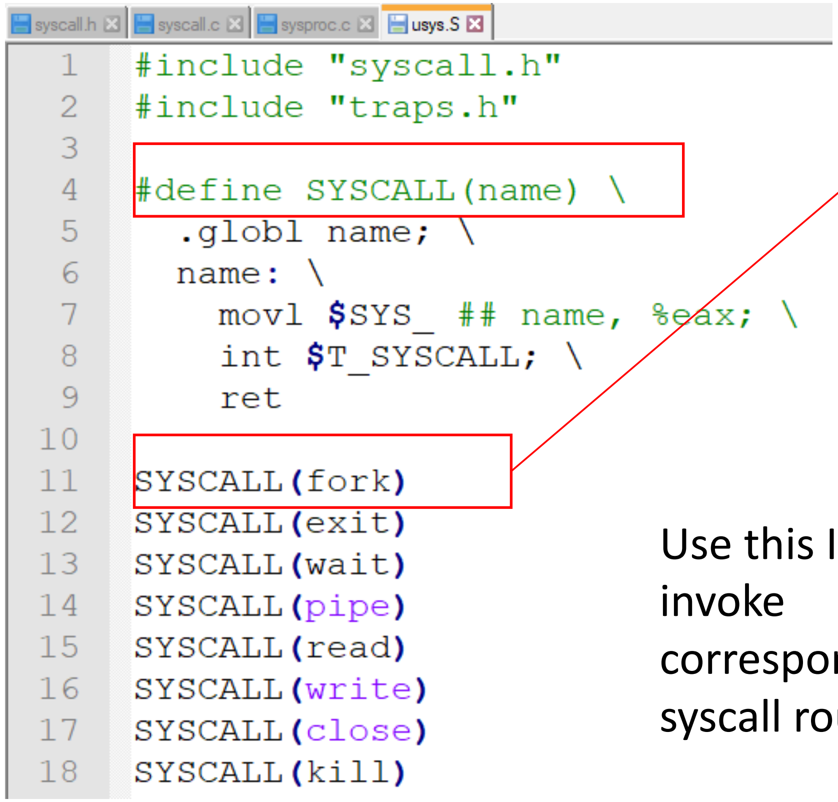
```
.globl fork; \
fork: \
  movl $SYS_fork, %eax; \
  int $T_SYSCALL; \
  ret
```

Move the syscall ID into eax register. The register value will be further pushed to stack (memory).

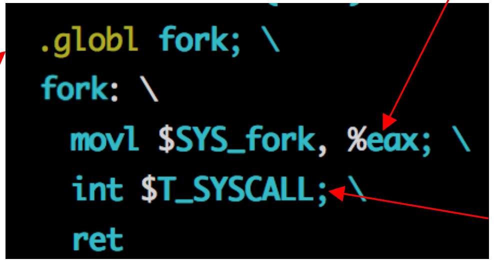
INT (interrupt) T_SYSCALL, which means there's a syscall request

CS 1550 – xv6 – Adding a custom Syscall

- Afterwards we define the interface for user programs to call
  - Open usys.S



```
#include "syscall.h"
#include "traps.h"

#define SYSCALL(name) \
  .globl name; \
  name: \
    movl $SYS_ ## name, %eax; \
    int $T_SYSCALL; \
    ret

SYSCALL(fork)
SYSCALL(exit)
SYSCALL(wait)
SYSCALL(pipe)
SYSCALL(read)
SYSCALL(write)
SYSCALL(close)
SYSCALL(kill)
```

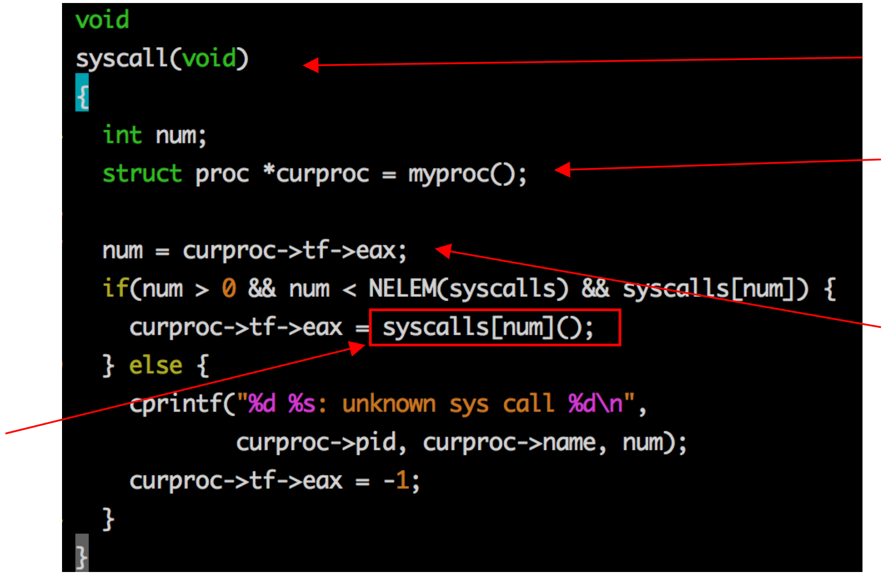


```
.globl fork; \
fork: \
  movl $SYS_fork, %eax; \
  int $T_SYSCALL; \
  ret
```

Move the syscall ID into eax register. The register value will be further pushed to stack (memory).

INT (interrupt) T_SYSCALL, which means there's a syscall request

In the end of syscall.c



```
void
syscall(void)
{
  int num;
  struct proc *curproc = myproc();

  num = curproc->tf->eax;
  if(num > 0 && num < NELEM(syscalls) && syscalls[num]) {
    curproc->tf->eax = syscalls[num]();
  } else {
    cprintf("%d %s: unknown sys call %d\n",
            curproc->pid, curproc->name, num);
    curproc->tf->eax = -1;
  }
}
```

Entrance of all syscall requests

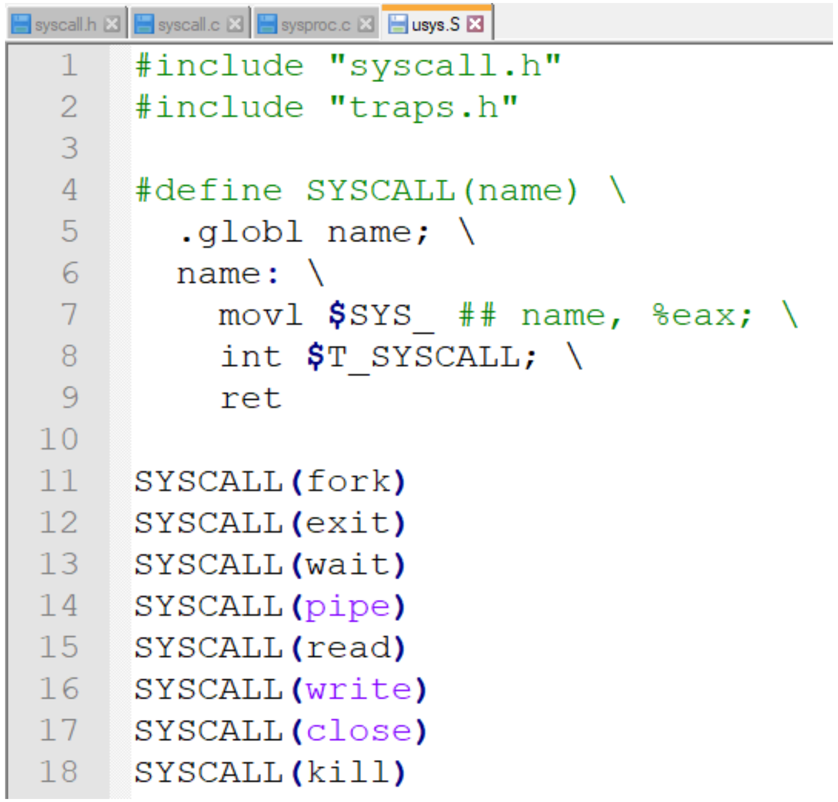
Each process has its own per-process state

Retrieve syscall ID (now it's already saved in the stack related to current process)

Use this ID to invoke corresponding syscall routine

CS 1550 – xv6 – Adding a custom Syscall

- Afterwards we define the interface for user programs to call
  - Open usys.S



```
#include "syscall.h"
#include "traps.h"

#define SYSCALL(name) \
  .globl name; \
  name: \
    movl $SYS_ ## name, %eax; \
    int $T_SYSCALL; \
    ret

SYSCALL(fork)
SYSCALL(exit)
SYSCALL(wait)
SYSCALL(pipe)
SYSCALL(read)
SYSCALL(write)
SYSCALL(close)
SYSCALL(kill)
```

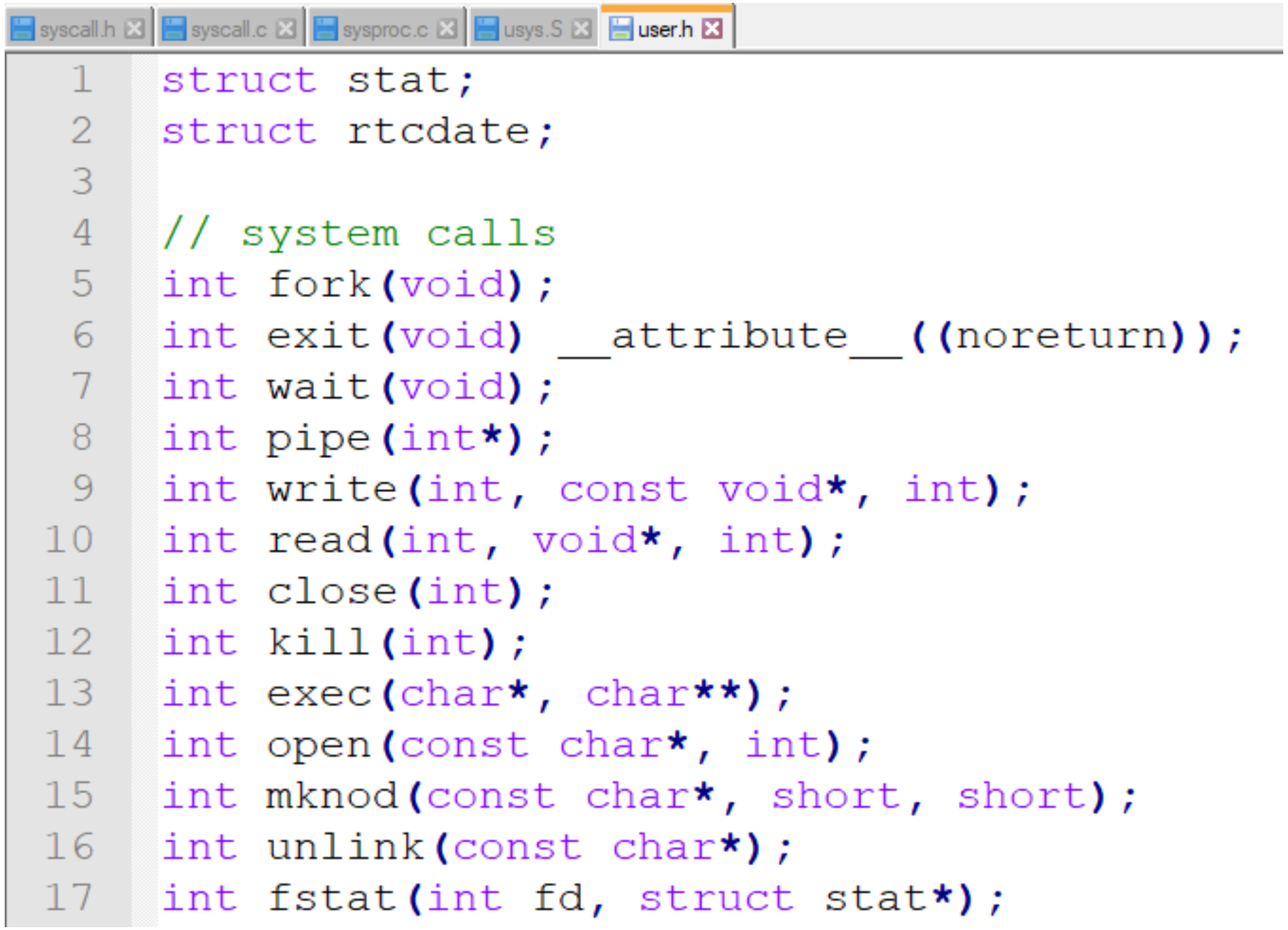
What you need to do in usys.S?

Add “SYSCALL(getday)”

CS 1550 – xv6 – Adding a custom Syscall

- Finally we open
  - user.h

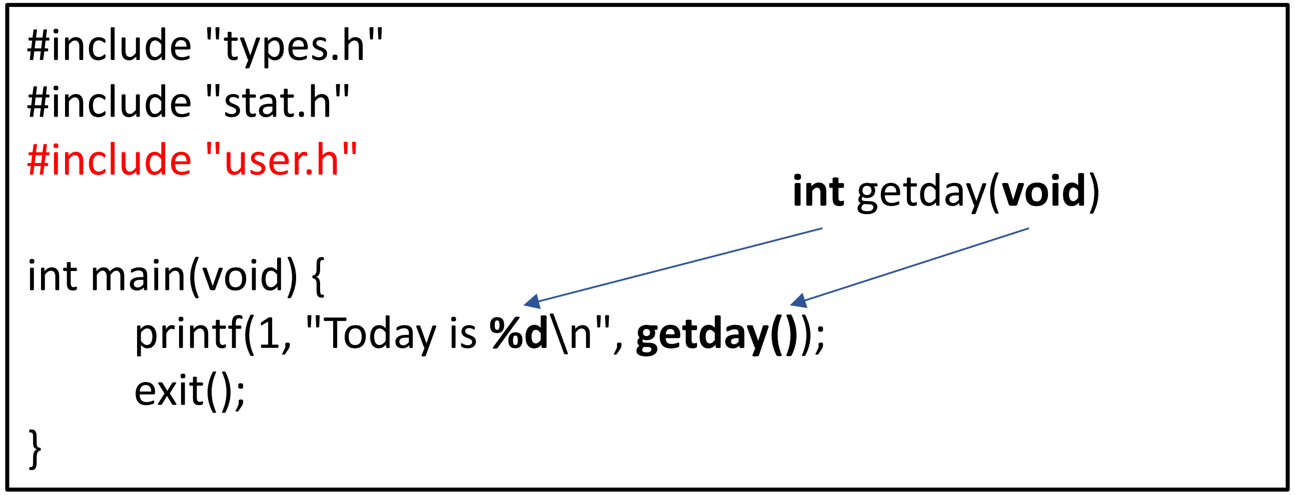
- Add “int getday(void);”



```
struct stat;
struct rtcdate;

// system calls
int fork(void);
int exit(void) __attribute__((noreturn));
int wait(void);
int pipe(int*);
int write(int, const void*, int);
int read(int, void*, int);
int close(int);
int kill(int);
int exec(char*, char**);
int open(const char*, int);
int mknod(const char*, short, short);
int unlink(const char*);
int fstat(int fd, struct stat*);
```

CS 1550 – xv6 – Adding a custom Syscall

- Write a user program for Xv6 to show today’s date
    - todays_date.c



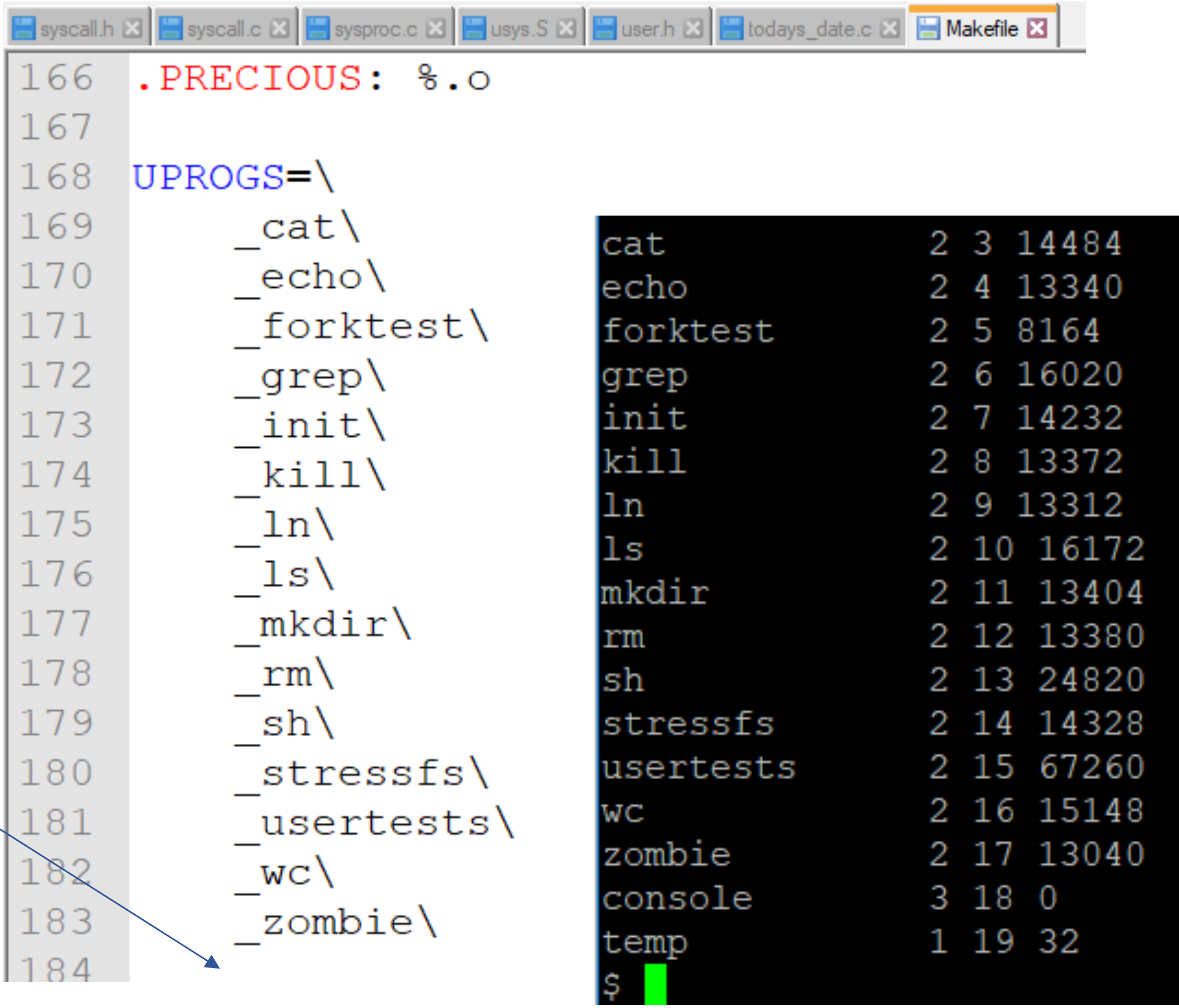
```
#include "types.h"
#include "stat.h"
#include "user.h"

int main(void) {
        printf(1, "Today is %d\n", getday());
        exit();
}
```

int getday(**void**)

CS 1550 – xv6 – Adding a custom Syscall

- Adding a user program, such that you can actually run it in XV6
  - Open Makefile

- Add
  - _todays_date\



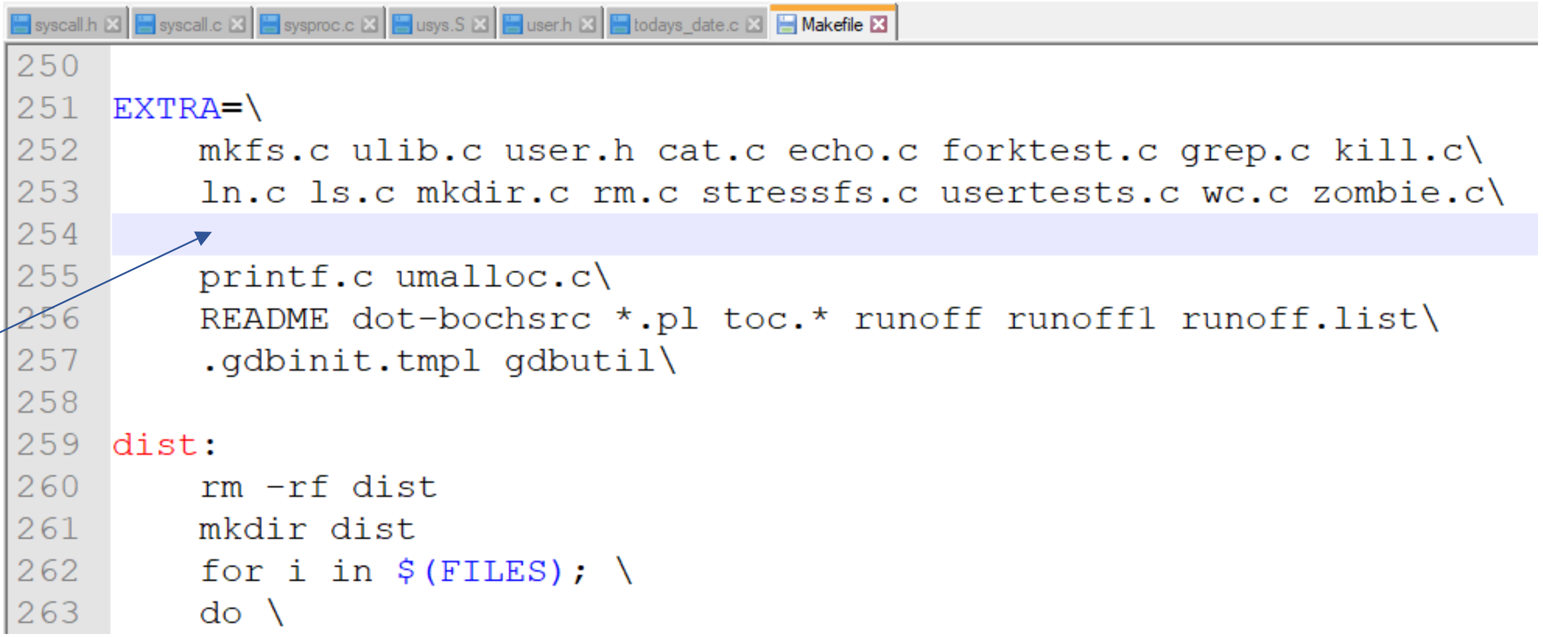
```
.PRECIOUS: %.o

UPROGS=\
	_cat\
	_echo\
	_forktest\
	_grep\
	_init\
	_kill\
	_ln\
	_ls\
	_mkdir\
	_rm\
	_sh\
	_stressfs\
	_usertests\
	_wc\
	_zombie\

```

```
cat            2 3 14484
echo           2 4 13340
forktest       2 5 8164
grep           2 6 16020
init           2 7 14232
kill           2 8 13372
ln             2 9 13312
ls             2 10 16172
mkdir          2 11 13404
rm             2 12 13380
sh             2 13 24820
stressfs       2 14 14328
usertests      2 15 67260
wc             2 16 15148
zombie         2 17 13040
console        3 18 0
temp           1 19 32
$
```

CS 1550 – xv6 – Adding a custom Syscall

- Adding a user program
  - Open Makefile

- and add
  - todays_date.c\



```

EXTRA=\
	mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\
	ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c\

	printf.c umalloc.c\
	README dot-bochsrc *.pl toc.* runoff runoff1 runoff.list\
	.gdbinit.tmpl gdbutil\

dist:
	rm -rf dist
	mkdir dist
	for i in $(FILES); \
	do \
```

CS 1550 – xv6 – Adding a custom Syscall

- Adding a user program
  - Open makefile
- and add
  - todays_date.c\
- Finally execute
  - make qemu-nox
- Run the program by calling it directly within xv6
  - todays_date

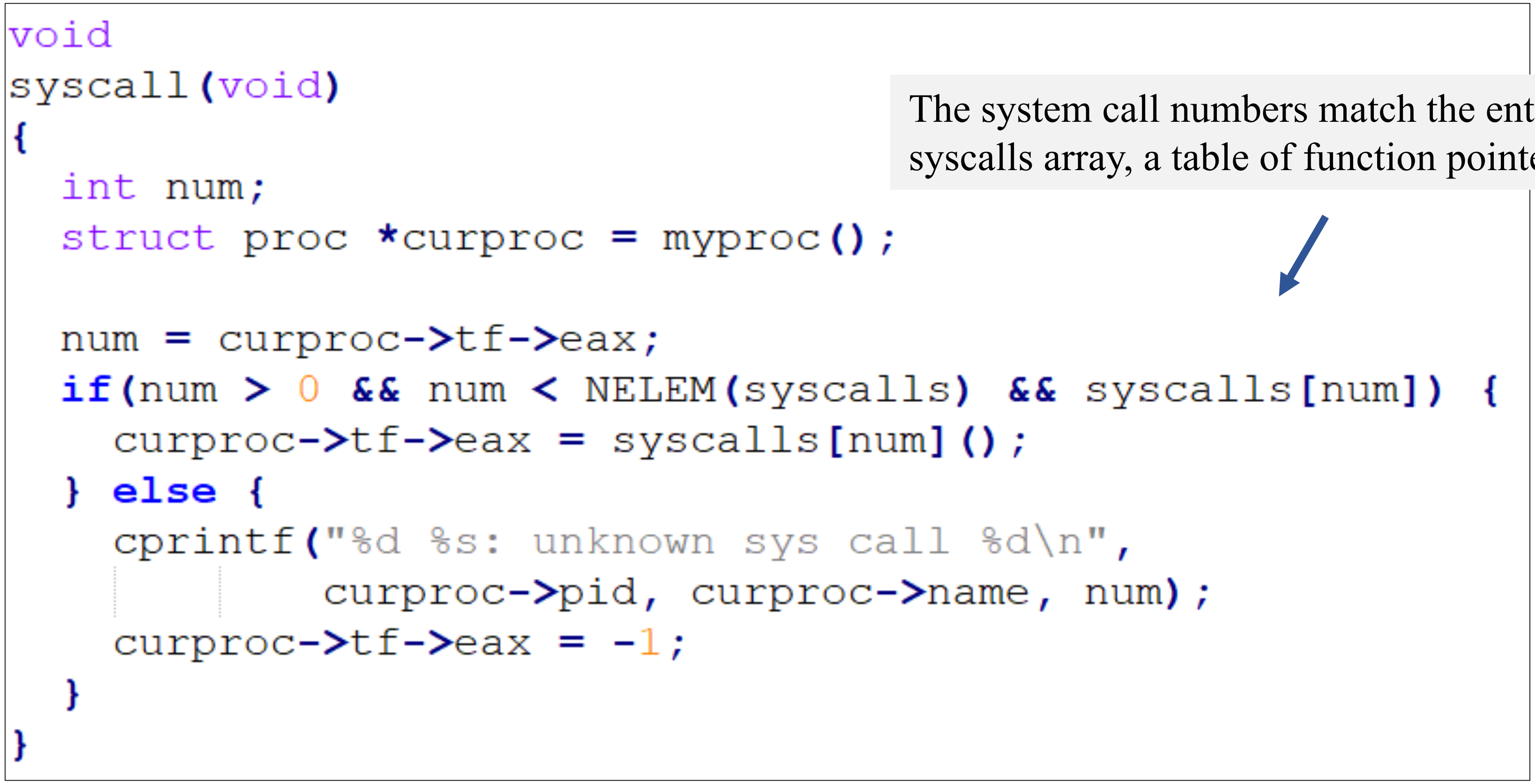
CS 1550 – xv6 exercise hints

- We need to worry about two things:
  - How to count syscalls invoked by the calling process?
  - Implement the method to return the count number given a syscall ID

CS 1550 – xv6 exercise hints

- Syscall calls will need a variable to hold the counting values
  - Where to write this data structure?
    - Which file holds processes metadata (per-process state)? proc.c, proc.h
  - The key function can be found in syscall.c
    - syscall(void) -> Is called every time any syscall is called

CS 1550 – xv6 exercise hints



```
void
syscall(void)
{
  int num;
  struct proc *curproc = myproc();

  num = curproc->tf->eax;
  if(num > 0 && num < NELEM(syscalls) && syscalls[num]) {
    curproc->tf->eax = syscalls[num]();
  } else {
    cprintf("%d %s: unknown sys call %d\n",
            curproc->pid, curproc->name, num);
    curproc->tf->eax = -1;
  }
}
```

The system call numbers match the entries in the syscalls array, a table of function pointers

CS 1550 – xv6 exercise hints

- Implementing **getcount**
  - Specify the method and its **id** in **syscall.h**
  - Specify extern method and pointer
    - **syscall.c**
  - Where to implement int **sys_getcount(void)**?
    - **sysproc.c**
  - Add SYSCALL(getcount)
    - **usys.S**
  - getcount.c
  - Modify proc.h, proc.c according to your method of counting.
    - Declare counting array? Hints: check proc.h to find the per-process state structure
    - Initialize counting array? Hints: check proc.c to find the function that performs initialization for processes

CS 1550 – xv6 exercise hints

- Submit to GradeScope the files that you have modified within the source code of xv6.
- You should modify the following files only:
  - syscall.h
  - syscall.c
  - user.h
  - usys.S
  - proc.h
  - proc.c
  - sysproc.c
  - Makefile