CSE2005
Operating Systems
Project J Component

Name: Dodda Sumanth & Kodeti Surya Vamsi
Project Supervisor: Prof.Padma Priya
Slot: L49+50

Reg.No: 18BCI0067&18BCI0080

Title:

ADDING SYSCALS IN XV6 OPERATING SYSTEM

About XV6:

It is an operating system based on linux kernel developed by MIT which is written in C and Assembly language and mainly it is an open source project which is free for everyone to modify and it is monolithic layered(entire operating system works in the kernel space) and it also has been deployed into many secure phones in China

Abstract:

Here,We are going to add a system call in to our XV6 operating system which is based on linux kernal and XV6 is an operating system and we are going to explain the process of adding a system

call in our kernal in detail and commands requires for adding a system call in our kernel

About PC used:

Here we are using Dual boot with Windows 10 and Ubuntu[18.04.2 LTS (bionic),64-bit] CPU info[Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz,800.058 Mhz]

What is a syscall:

In computing, a system call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. This may include hardware related services, creation and execution of new processes, and communication with integral kernel services such as process scheduling.

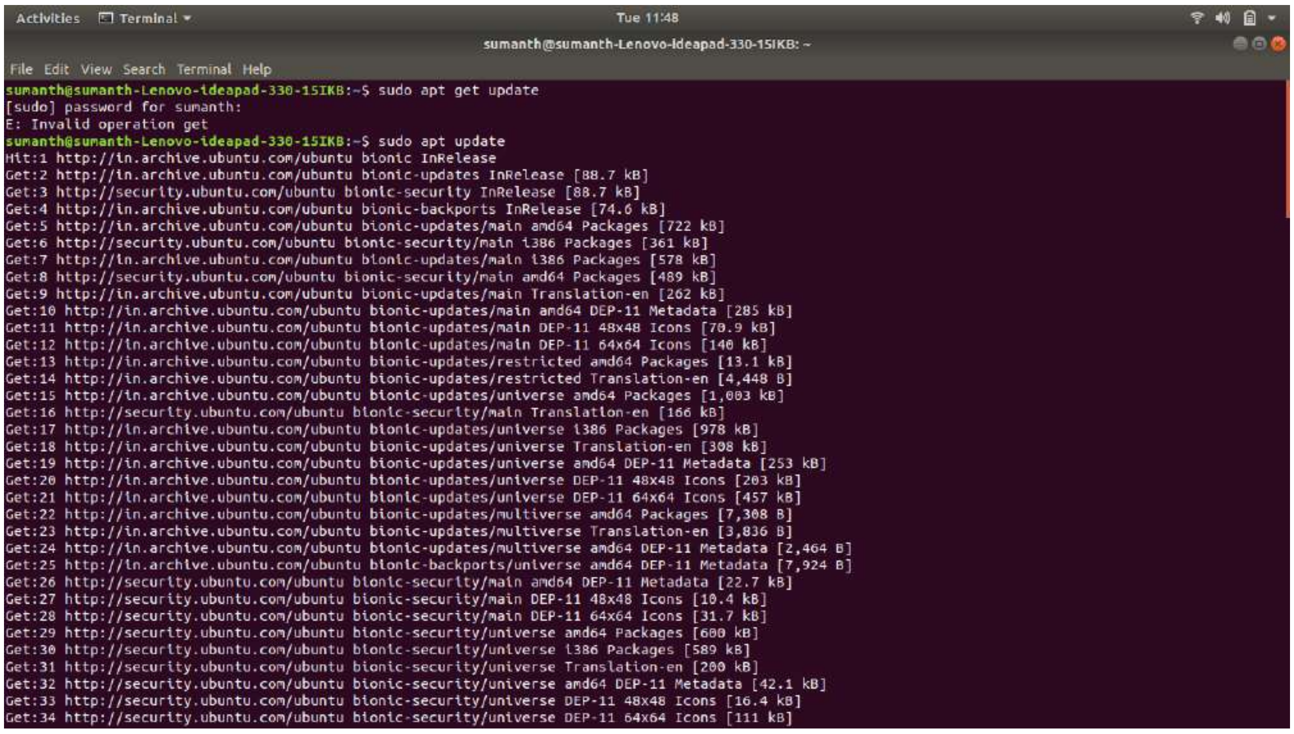
In computing, a system call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to interact with the operating system. A computer program makes a system call when it makes a request to the operating system's kernel.

Introduction about Installation and modules used:

Before we are going to install anything we need to keep our Ubuntu updated because updates make our work eaiser with new patches and also helps us to keep our pc secured!

Updating:

```
$ sudo apt-get update
```


Activities
Terminal
Tue 11:48
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~
File Edit View Search Terminal Help

```
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$ sudo apt get update
[sudo] password for sumanth:
E: Invalid operation get
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$ sudo apt update
Hit:1 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [722 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security/main i386 Packages [361 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu bionic-updates/main i386 Packages [578 kB]
Get:8 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [489 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu bionic-updates/main Translation-en [262 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu bionic-updates/main amd64 DEP-11 Metadata [285 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu bionic-updates/main DEP-11 48x48 Icons [70.9 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu bionic-updates/main DEP-11 64x64 Icons [140 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [13.1 kB]
Get:14 http://in.archive.ubuntu.com/ubuntu bionic-updates/restricted Translation-en [4,448 B]
Get:15 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1,003 kB]
Get:16 http://security.ubuntu.com/ubuntu bionic-security/main Translation-en [166 kB]
Get:17 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe i386 Packages [978 kB]
Get:18 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe Translation-en [308 kB]
Get:19 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 DEP-11 Metadata [253 kB]
Get:20 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe DEP-11 48x48 Icons [203 kB]
Get:21 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe DEP-11 64x64 Icons [457 kB]
Get:22 http://in.archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [7,308 B]
Get:23 http://in.archive.ubuntu.com/ubuntu bionic-updates/multiverse Translation-en [3,836 B]
Get:24 http://in.archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 DEP-11 Metadata [2,464 B]
Get:25 http://in.archive.ubuntu.com/ubuntu bionic-backports/universe amd64 DEP-11 Metadata [7,924 B]
Get:26 http://security.ubuntu.com/ubuntu bionic-security/main amd64 DEP-11 Metadata [22.7 kB]
Get:27 http://security.ubuntu.com/ubuntu bionic-security/main DEP-11 48x48 Icons [10.4 kB]
Get:28 http://security.ubuntu.com/ubuntu bionic-security/main DEP-11 64x64 Icons [31.7 kB]
Get:29 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [600 kB]
Get:30 http://security.ubuntu.com/ubuntu bionic-security/universe i386 Packages [589 kB]
Get:31 http://security.ubuntu.com/ubuntu bionic-security/universe Translation-en [200 kB]
Get:32 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 DEP-11 Metadata [42.1 kB]
Get:33 http://security.ubuntu.com/ubuntu bionic-security/universe DEP-11 48x48 Icons [16.4 kB]
Get:34 http://security.ubuntu.com/ubuntu bionic-security/universe DEP-11 64x64 Icons [111 kB]
```

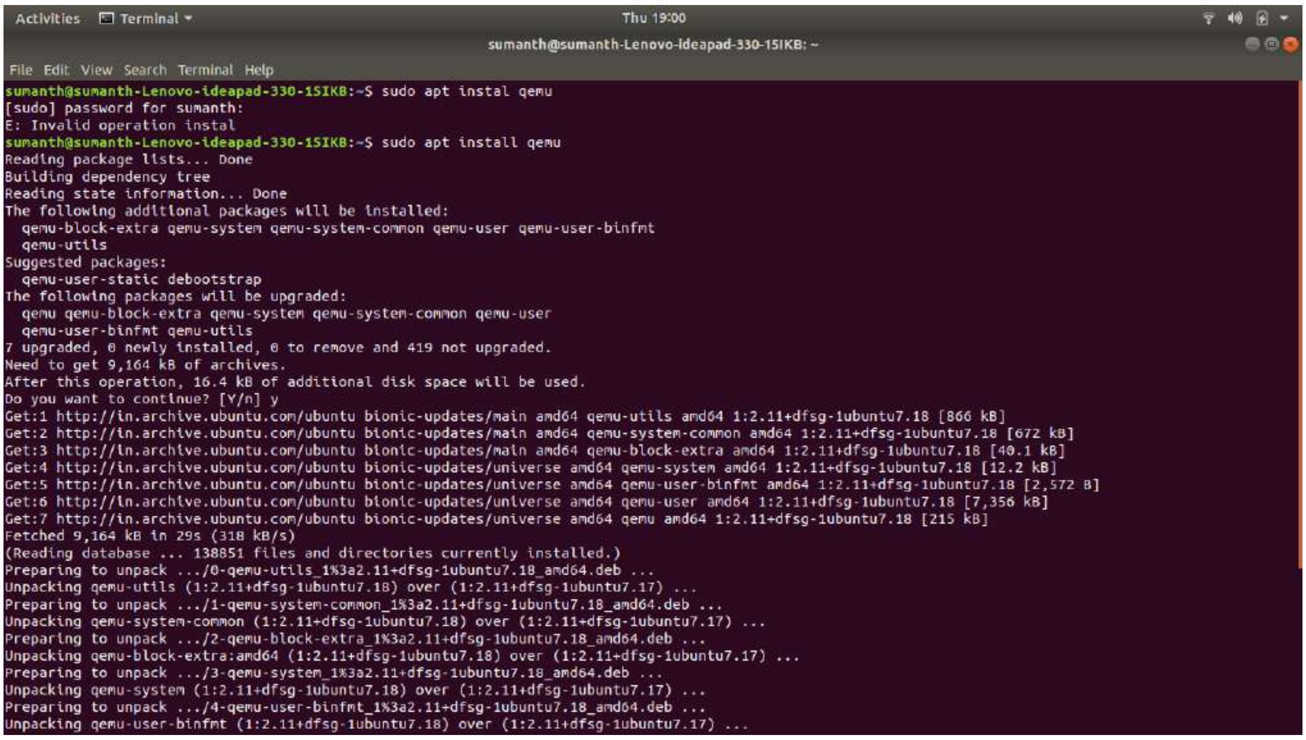
We are going to install 2 things :

1. qemu
2. git repository(clone with git clone <url>.)
3. XV6(git://github.com/mit-pdos/xv6-public.git)
4. make repository(To determine automatically which pieces of a large program need to be re-compiled, and issue the commands necessary to recompile them)

What is qemu?
QEMU stands for Quick Emulator and what it does is it emulates the machine's processor through dynamic binary translation(where sequences of instructions are translated from a source instruction to the target instruction set) and provides a set of different hardware and device models for the machine, enabling it to run a variety of operating systems like XV6

Installing Qemu:

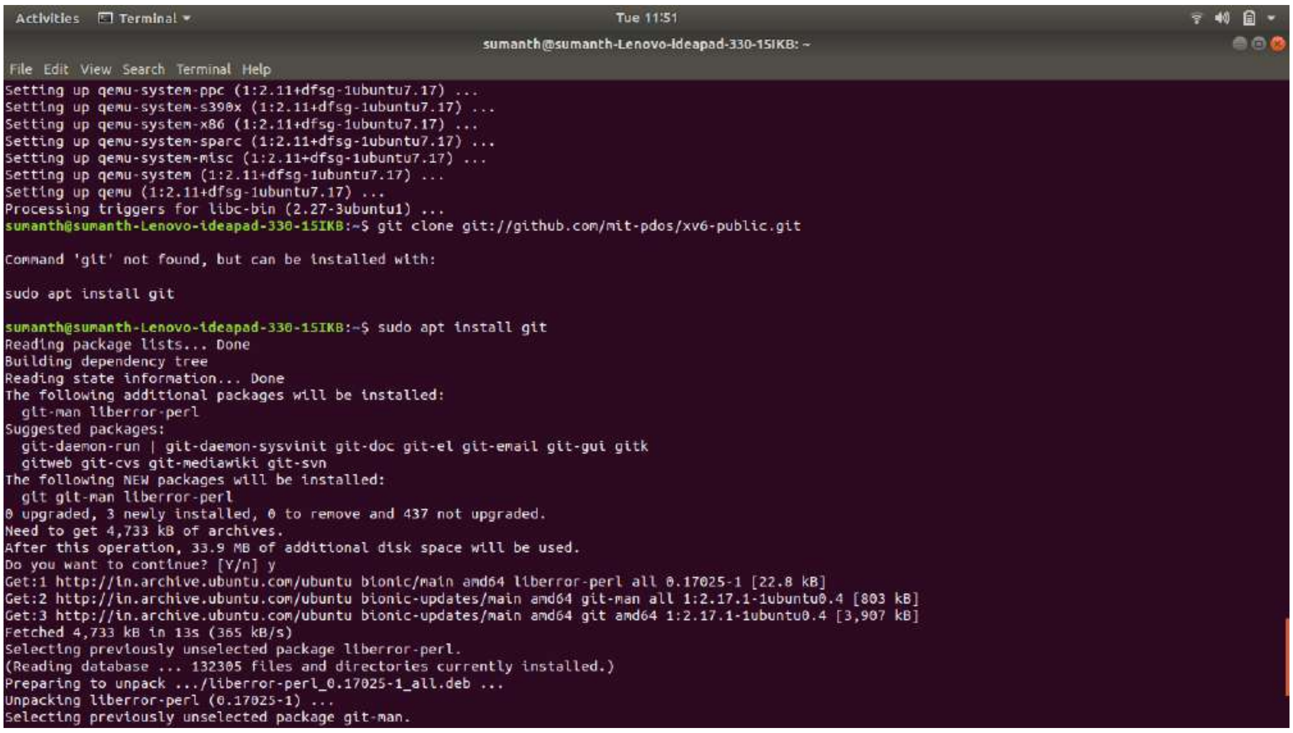
```
$ sudo apt install qemu
```


Activities
Terminal
Thu 19:00
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~
File Edit View Search Terminal Help

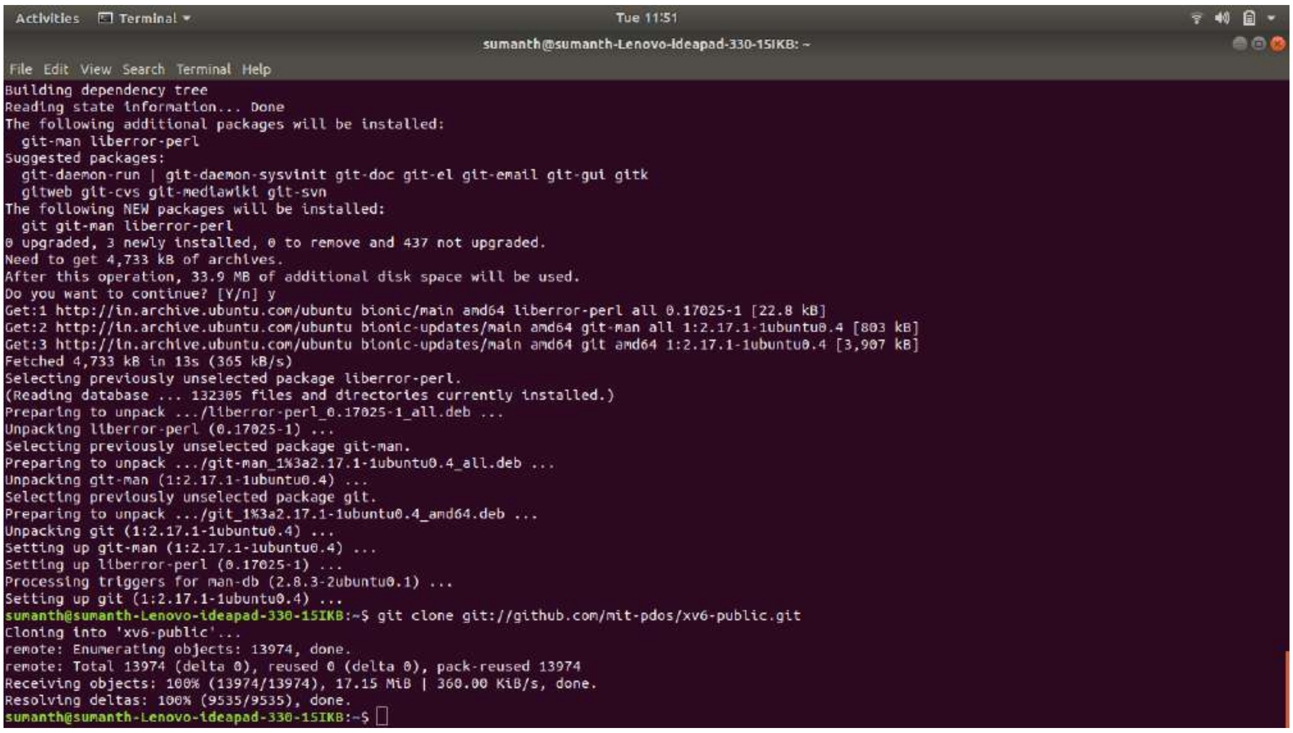
```
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$ sudo apt instal qemu
[sudo] password for sumanth:
E: Invalid operation instal
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$ sudo apt install qemu
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  qemu-block-extra qemu-system qemu-system-common qemu-user qemu-user-binfmt
  qemu-utils
Suggested packages:
  qemu-user-static debootstrap
The following packages will be upgraded:
  qemu qemu-block-extra qemu-system qemu-system-common qemu-user
  qemu-user-binfmt qemu-utils
7 upgraded, 0 newly installed, 0 to remove and 419 not upgraded.
Need to get 9,164 kB of archives.
After this operation, 16.4 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu bionic-updates/main amd64 qemu-utils amd64 1:2.11+dfsg-1ubuntu7.18 [866 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu bionic-updates/main amd64 qemu-system-common amd64 1:2.11+dfsg-1ubuntu7.18 [672 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu bionic-updates/main amd64 qemu-block-extra amd64 1:2.11+dfsg-1ubuntu7.18 [40.1 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 qemu-system amd64 1:2.11+dfsg-1ubuntu7.18 [12.2 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 qemu-user-binfmt amd64 1:2.11+dfsg-1ubuntu7.18 [2,572 B]
Get:6 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 qemu-user amd64 1:2.11+dfsg-1ubuntu7.18 [7,356 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 qemu amd64 1:2.11+dfsg-1ubuntu7.18 [215 kB]
Fetched 9,164 kB in 29s (318 kB/s)
(Reading database ... 138851 files and directories currently installed.)
Preparing to unpack .../0-qemu-utils_1%3a2.11+dfsg-1ubuntu7.18_amd64.deb ...
Unpacking qemu-utils (1:2.11+dfsg-1ubuntu7.18) over (1:2.11+dfsg-1ubuntu7.17) ...
Preparing to unpack .../1-qemu-system-common_1%3a2.11+dfsg-1ubuntu7.18_amd64.deb ...
Unpacking qemu-system-common (1:2.11+dfsg-1ubuntu7.18) over (1:2.11+dfsg-1ubuntu7.17) ...
Preparing to unpack .../2-qemu-block-extra_1%3a2.11+dfsg-1ubuntu7.18_amd64.deb ...
Unpacking qemu-block-extra:amd64 (1:2.11+dfsg-1ubuntu7.18) over (1:2.11+dfsg-1ubuntu7.17) ...
Preparing to unpack .../3-qemu-system_1%3a2.11+dfsg-1ubuntu7.18_amd64.deb ...
Unpacking qemu-system (1:2.11+dfsg-1ubuntu7.18) over (1:2.11+dfsg-1ubuntu7.17) ...
Preparing to unpack .../4-qemu-user-binfmt_1%3a2.11+dfsg-1ubuntu7.18_amd64.deb ...
Unpacking qemu-user-binfmt (1:2.11+dfsg-1ubuntu7.18) over (1:2.11+dfsg-1ubuntu7.17) ...
```

Installing git repository to clone XV6 from github:

$ sudo apt install git


Activities
Terminal ▾
Tue 11:51
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~
File Edit View Search Terminal Help

```
Setting up qemu-system-ppc (1:2.11+dfsg-1ubuntu7.17) ...
Setting up qemu-system-s390x (1:2.11+dfsg-1ubuntu7.17) ...
Setting up qemu-system-x86 (1:2.11+dfsg-1ubuntu7.17) ...
Setting up qemu-system-sparc (1:2.11+dfsg-1ubuntu7.17) ...
Setting up qemu-system-misc (1:2.11+dfsg-1ubuntu7.17) ...
Setting up qemu-system (1:2.11+dfsg-1ubuntu7.17) ...
Setting up qemu (1:2.11+dfsg-1ubuntu7.17) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$ git clone git://github.com/mit-pdos/xv6-public.git

Command 'git' not found, but can be installed with:

sudo apt install git

sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$ sudo apt install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 437 not upgraded.
Need to get 4,733 kB of archives.
After this operation, 33.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu bionic/main amd64 liberror-perl all 0.17025-1 [22.8 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git-man all 1:2.17.1-1ubuntu0.4 [803 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git amd64 1:2.17.1-1ubuntu0.4 [3,907 kB]
Fetched 4,733 kB in 13s (365 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 132305 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17025-1_all.deb ...
Unpacking liberror-perl (0.17025-1) ...
Selecting previously unselected package git-man.
```


Activities
Terminal ▾
Tue 11:51
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~
File Edit View Search Terminal Help

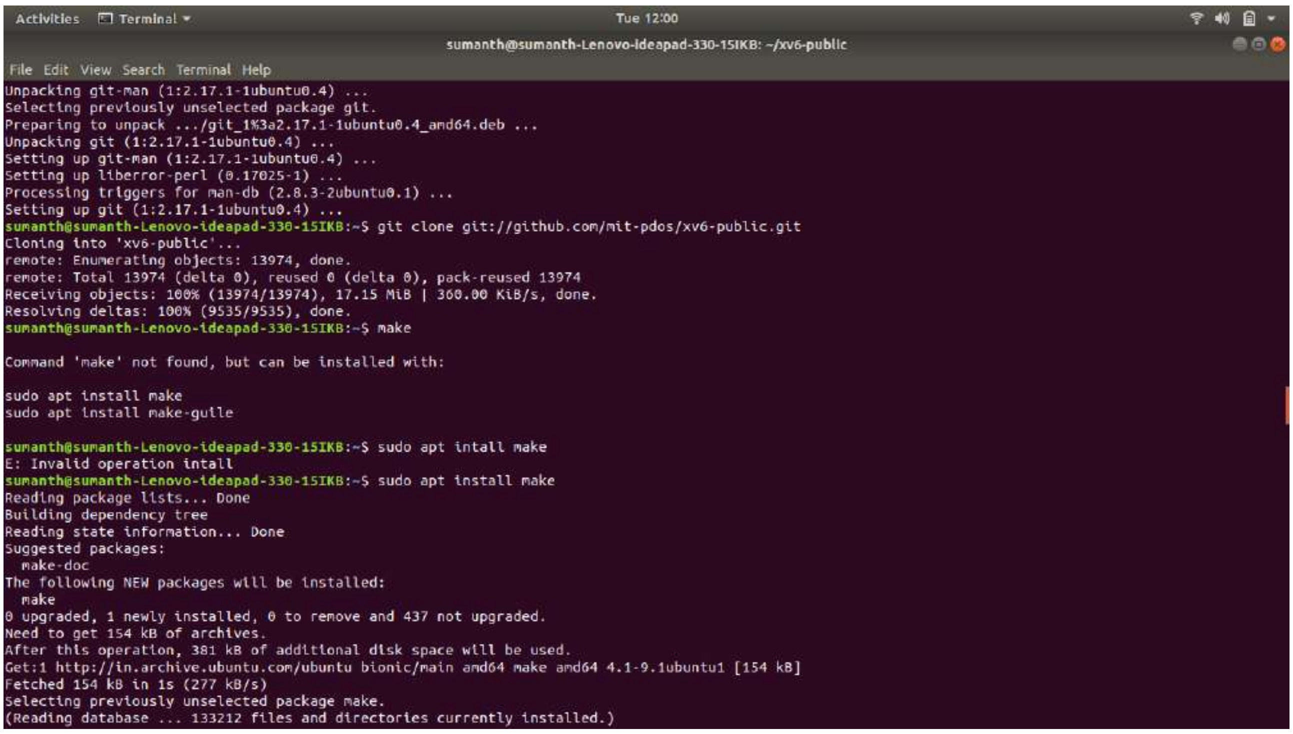
```
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 437 not upgraded.
Need to get 4,733 kB of archives.
After this operation, 33.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu bionic/main amd64 liberror-perl all 0.17025-1 [22.8 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git-man all 1:2.17.1-1ubuntu0.4 [803 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git amd64 1:2.17.1-1ubuntu0.4 [3,907 kB]
Fetched 4,733 kB in 13s (365 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 132305 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17025-1_all.deb ...
Unpacking liberror-perl (0.17025-1) ...
Selecting previously unselected package git-man.
Preparing to unpack .../git-man_1%3a2.17.1-1ubuntu0.4_all.deb ...
Unpacking git-man (1:2.17.1-1ubuntu0.4) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.17.1-1ubuntu0.4_amd64.deb ...
Unpacking git (1:2.17.1-1ubuntu0.4) ...
Setting up git-man (1:2.17.1-1ubuntu0.4) ...
Setting up liberror-perl (0.17025-1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Setting up git (1:2.17.1-1ubuntu0.4) ...
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$ git clone git://github.com/mit-pdos/xv6-public.git
Cloning into 'xv6-public'...
remote: Enumerating objects: 13974, done.
remote: Total 13974 (delta 0), reused 0 (delta 0), pack-reused 13974
Receiving objects: 100% (13974/13974), 17.15 MiB | 360.00 KiB/s, done.
Resolving deltas: 100% (9535/9535), done.
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$
```

Using git and cloning XV6 OS from git://github.com/mit-pdos/xv6-public.git :

$ git clone git://github.com/mit-pdos/xv6-public.git

Installing make repository:

$ sudo apt install make

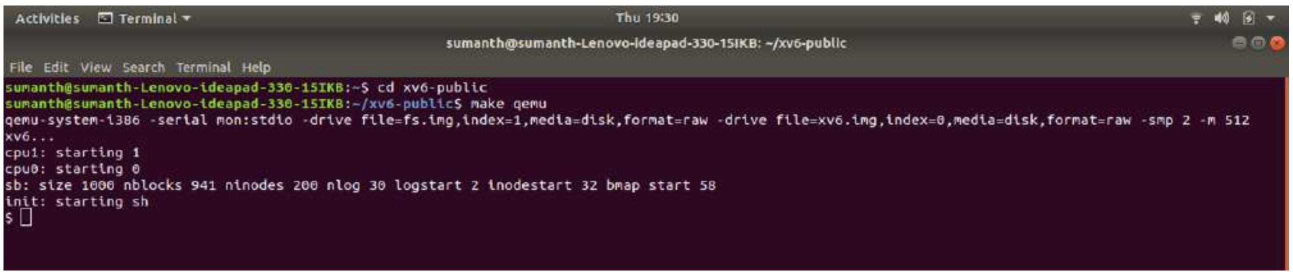


Activities
Terminal
Tue 12:00
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
Unpacking git-man (1:2.17.1-1ubuntu0.4) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.17.1-1ubuntu0.4_amd64.deb ...
Unpacking git (1:2.17.1-1ubuntu0.4) ...
Setting up git-man (1:2.17.1-1ubuntu0.4) ...
Setting up liberror-perl (0.17025-1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Setting up git (1:2.17.1-1ubuntu0.4) ...
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$ git clone git://github.com/mit-pdos/xv6-public.git
Cloning into 'xv6-public'...
remote: Enumerating objects: 13974, done.
remote: Total 13974 (delta 0), reused 0 (delta 0), pack-reused 13974
Receiving objects: 100% (13974/13974), 17.15 MiB | 360.00 KiB/s, done.
Resolving deltas: 100% (9535/9535), done.
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$ make

Command 'make' not found, but can be installed with:

sudo apt install make
sudo apt install make-guile

sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$ sudo apt intall make
E: Invalid operation intall
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$ sudo apt install make
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  make-doc
The following NEW packages will be installed:
  make
0 upgraded, 1 newly installed, 0 to remove and 437 not upgraded.
Need to get 154 kB of archives.
After this operation, 381 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu bionic/main amd64 make amd64 4.1-9.1ubuntu1 [154 kB]
Fetched 154 kB in 1s (277 kB/s)
Selecting previously unselected package make.
(Reading database ... 133212 files and directories currently installed.)
```

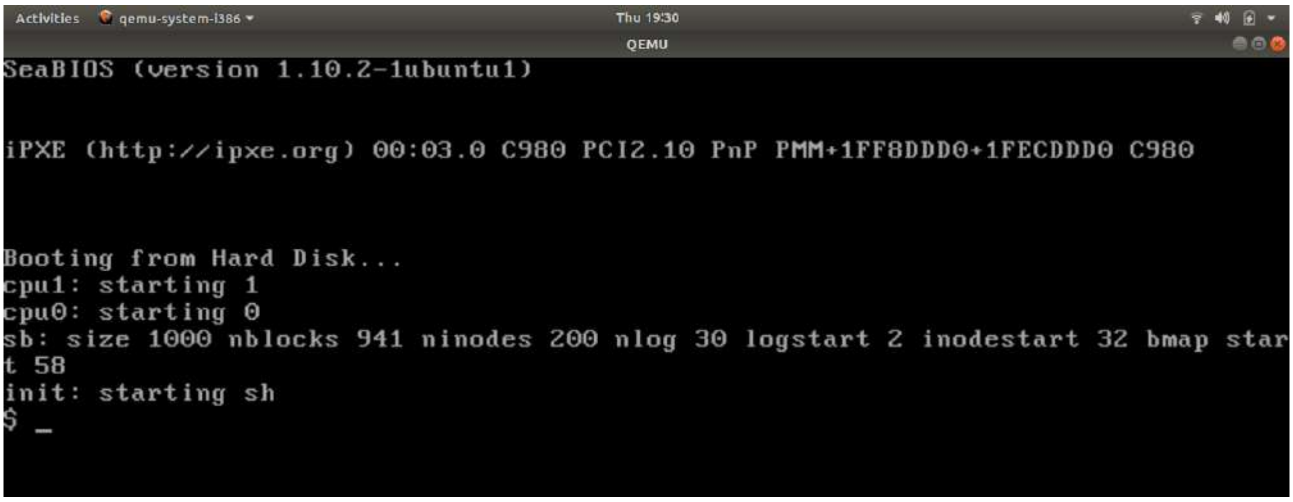
Now we came to know that we need to install xv6-public directory and we have to give the command $make qemu to start our xv6 OS

Running XV6:



Activities
Terminal
Thu 19:30
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$ cd xv6-public
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ make qemu
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ 
```



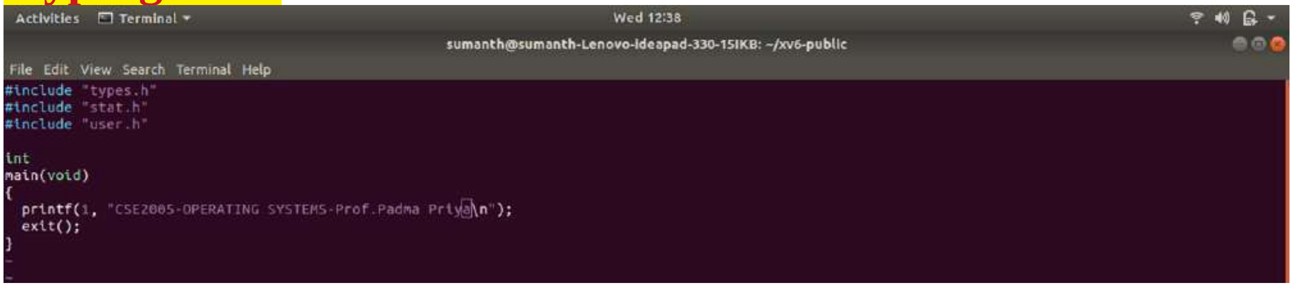
```
SeaBIOS (version 1.10.2-1ubuntu1)


iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF8DDD0+1FECDDD0 C980



Booting from Hard Disk...
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
$ _
```

Introduction about the concept/methodology taken

We are going to add system call to our operating system the priority for system will be the very high rather than all the processes and it interrupts all the processes and alits the CPU to the syscall which is called by the user

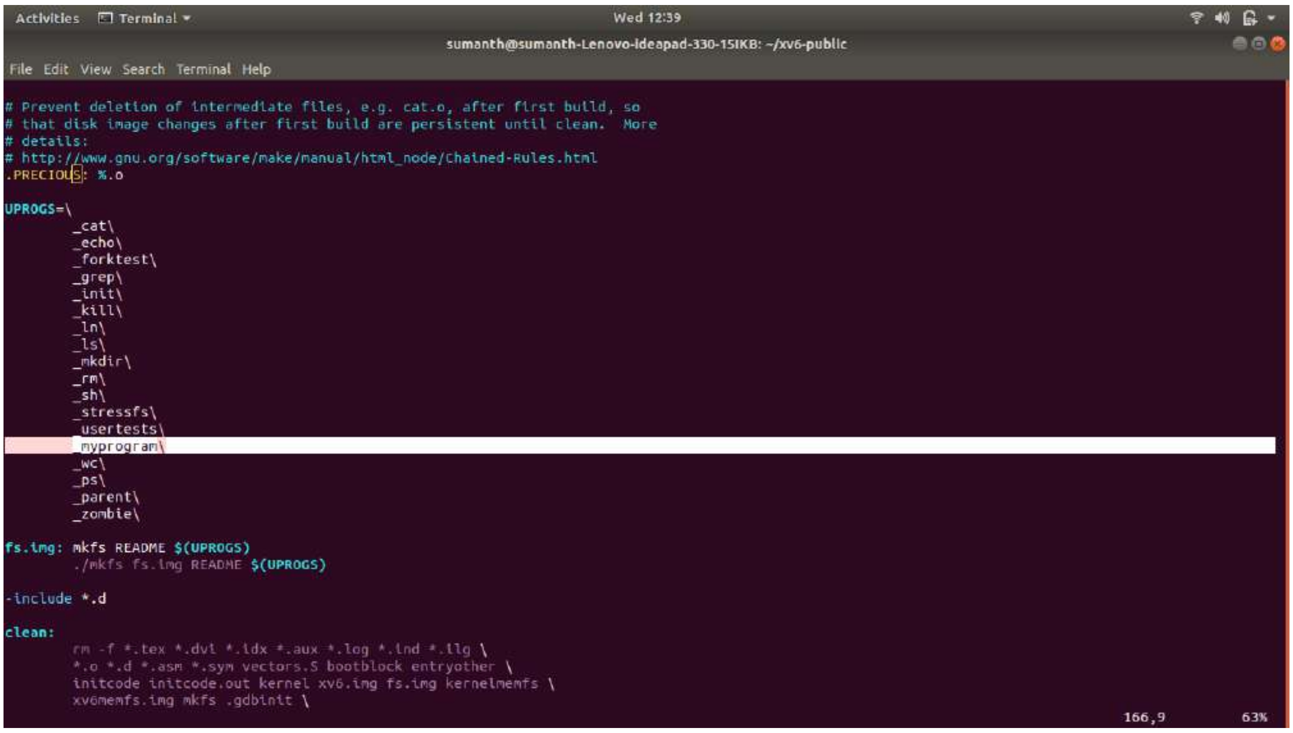
In computing, a system call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. This may include hardware-related services (for example, accessing a hard disk drive), creation and execution of new processes, and communication with integral kernel services such as process scheduling. System calls provide an essential interface between a process and the operating system.

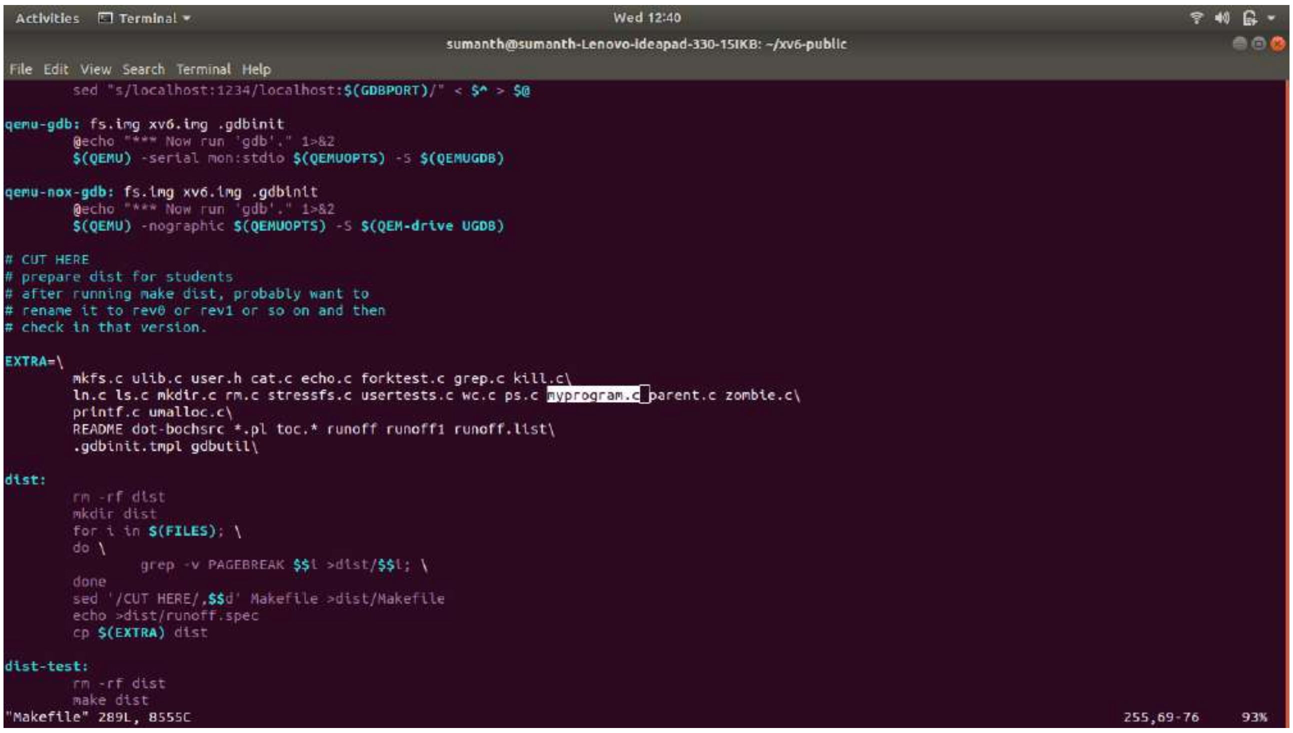
Creating
myprogram.c



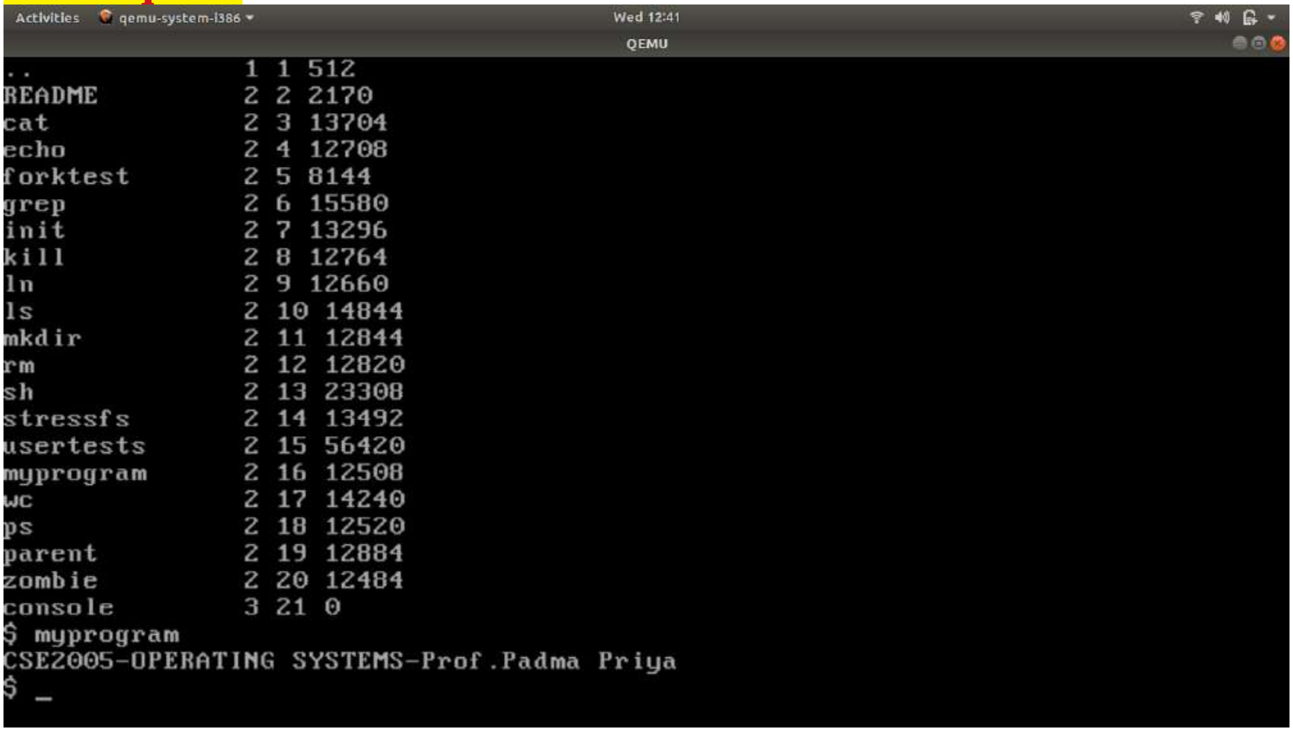
```
#include "types.h"
#include "stat.h"
#include "user.h"

int
main(void)
{
  printf(1, "CSE2005-OPERATING SYSTEMS-Prof.Padma Priya\n");
  exit();
}
~
~
```

Makefile


```
# Prevent deletion of intermediate files, e.g. cat.o, after first build, so
# that disk image changes after first build are persistent until clean.  More
# details:
# http://www.gnu.org/software/make/manual/html_node/Chained-Rules.html
.PRECIOUS: %.o

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

fs.img: mkfs README $(UPROGS)
	./mkfs fs.img README $(UPROGS)

-include *.d

clean:
	rm -f *.tex *.dvi *.idx *.aux *.log *.ind *.ilg \
	*.o *.d *.asm *.sym vectors.S bootblock entryother \
	initcode initcode.out kernel xv6.img fs.img kernelmemfs \
	xv6memfs.img mkfs .gdbinit \
                                                                         166,9         63%
```


```
	sed "s/localhost:1234/localhost:$(GDBPORT)/" < $^ > $@

qemu-gdb: fs.img xv6.img .gdbinit
	@echo "*** Now run 'gdb'." 1>&2
	$(QEMU) -serial mon:stdio $(QEMUOPTS) -S $(QEMUGDB)

qemu-nox-gdb: fs.img xv6.img .gdbinit
	@echo "*** Now run 'gdb'." 1>&2
	$(QEMU) -nographic $(QEMUOPTS) -S $(QEM-drive UGDB)

# CUT HERE
# prepare dist for students
# after running make dist, probably want to
# rename it to rev0 or rev1 or so on and then
# check in that version.

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 ps.c myprogram.c parent.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 \
		grep -v PAGEBREAK $$i >dist/$$i; \
	done
	sed '/CUT HERE/,$$d' Makefile >dist/Makefile
	echo >dist/runoff.spec
	cp $(EXTRA) dist

dist-test:
	rm -rf dist
	make dist
"Makefile" 289L, 8555C                                                    255,69-76     93%
```

make qemu

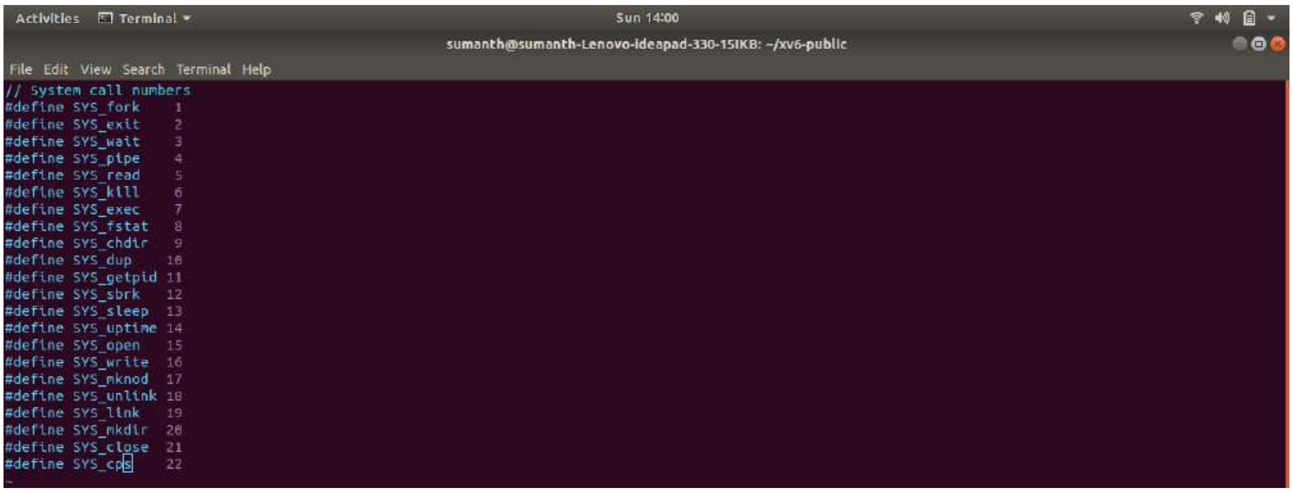


```
..             1 1 512
README         2 2 2170
cat            2 3 13704
echo           2 4 12708
forktest       2 5 8144
grep           2 6 15580
init           2 7 13296
kill           2 8 12764
ln             2 9 12660
ls             2 10 14844
mkdir          2 11 12844
rm             2 12 12820
sh             2 13 23308
stressfs       2 14 13492
usertests      2 15 56420
myprogram      2 16 12508
wc             2 17 14240
ps             2 18 12520
parent         2 19 12884
zombie         2 20 12484
console        3 21 0
$ myprogram
CSE2005-OPERATING SYSTEMS-Prof.Padma Priya
$ _
```

Modules Shown in previous reviews:

1. syscall.h

This defines the position of the system call vector that connects to the implementation.

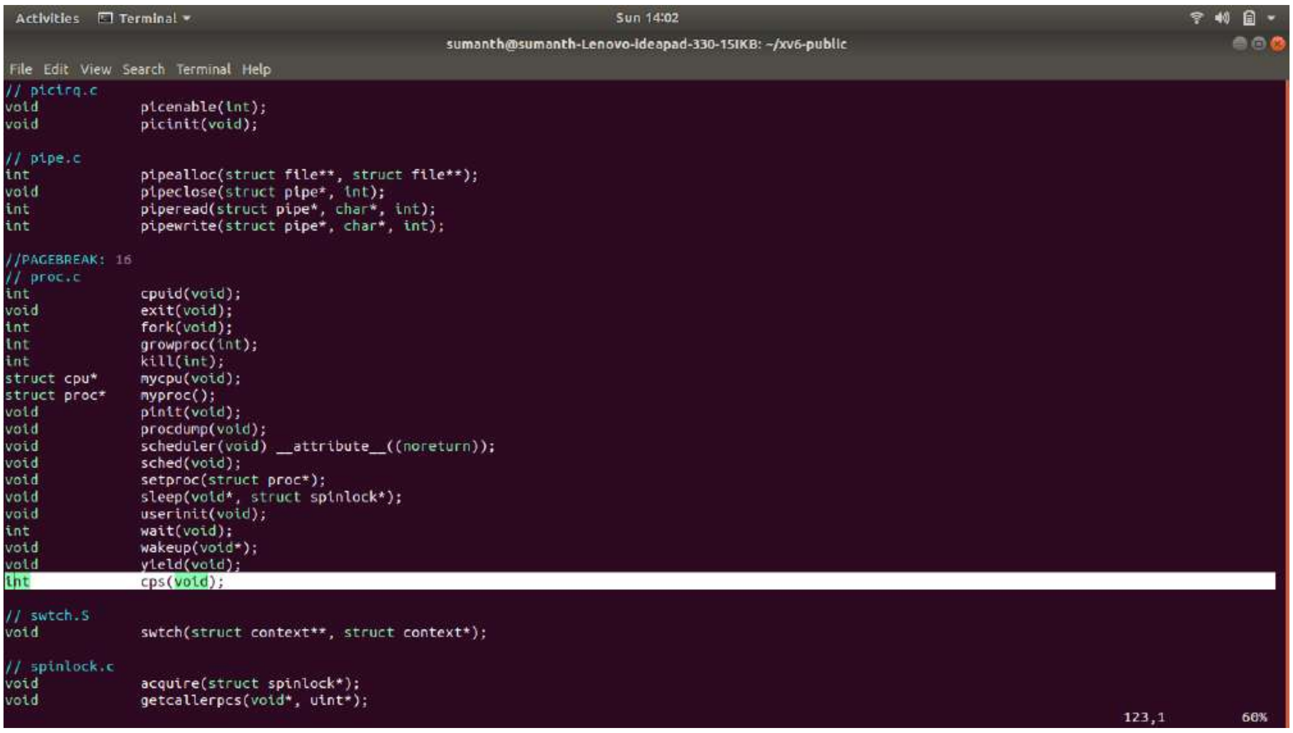

Activities
Terminal ▾
Sun 14:00
sumanth@sumanth-Lenovo-Ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
// 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
#define SYS_unlink 18
#define SYS_link   19
#define SYS_mkdir  20
#define SYS_close  21
#define SYS_cps    22
~
```

2. defs.h

This adds a forward declaration for the new system call.

We add this function in proc.c

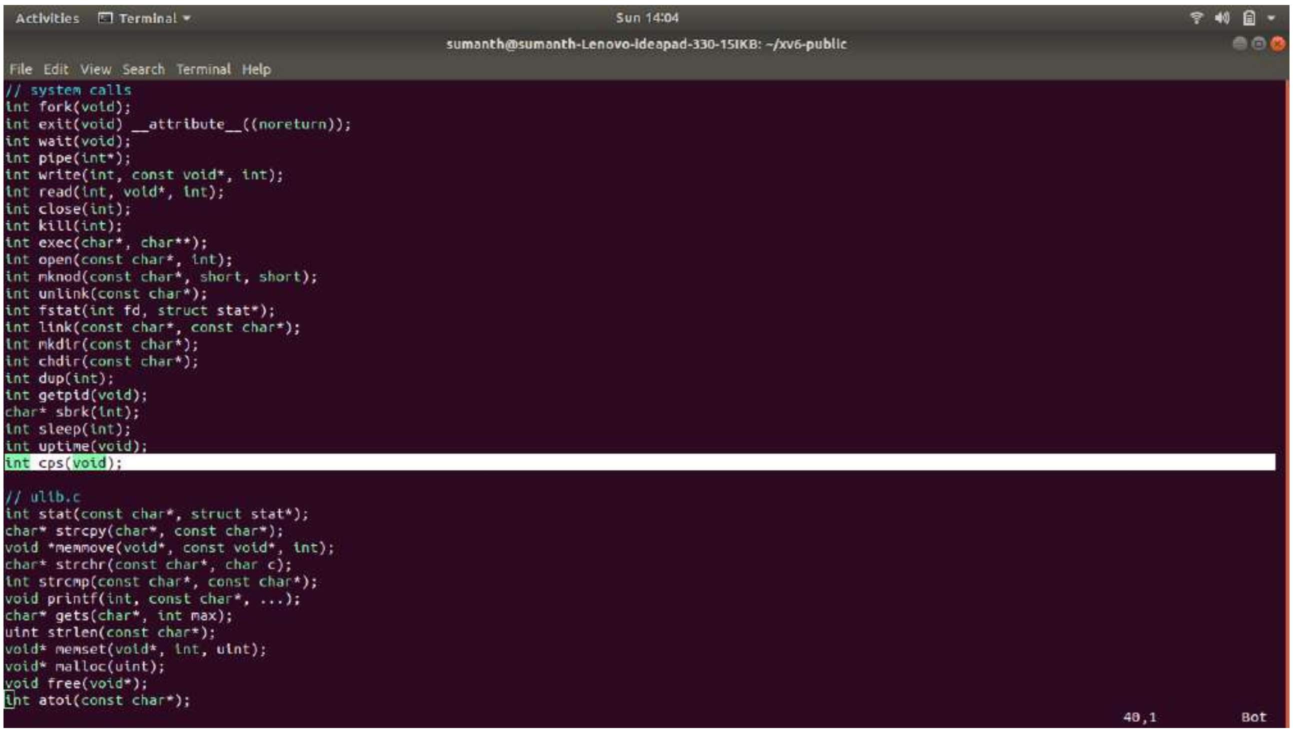

Activities
Terminal ▾
Sun 14:02
sumanth@sumanth-Lenovo-Ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
// picirq.c
void            picenable(int);
void            picinit(void);

// pipe.c
int             pipealloc(struct file**, struct file**);
void            pipeclose(struct pipe*, int);
int             piperead(struct pipe*, char*, int);
int             pipewrite(struct pipe*, char*, int);

//PAGEBREAK: 16
// proc.c
int             cpuid(void);
void            exit(void);
int             fork(void);
int             growproc(int);
int             kill(int);
struct cpu*     mycpu(void);
struct proc*    myproc();
void            pinit(void);
void            procdump(void);
void            scheduler(void) __attribute__((noreturn));
void            sched(void);
void            setproc(struct proc*);
void            sleep(void*, struct spinlock*);
void            userinit(void);
int             wait(void);
void            wakeup(void*);
void            yield(void);
int             cps(void);

// swtch.S
void            swtch(struct context**, struct context*);

// spinlock.c
void            acquire(struct spinlock*);
void            getcallerpcs(void*, uint*);
                                                                123,1          60%
```

3. user.h

It defines the function that can be called through the shell.

We add this function prototype in syscalls.

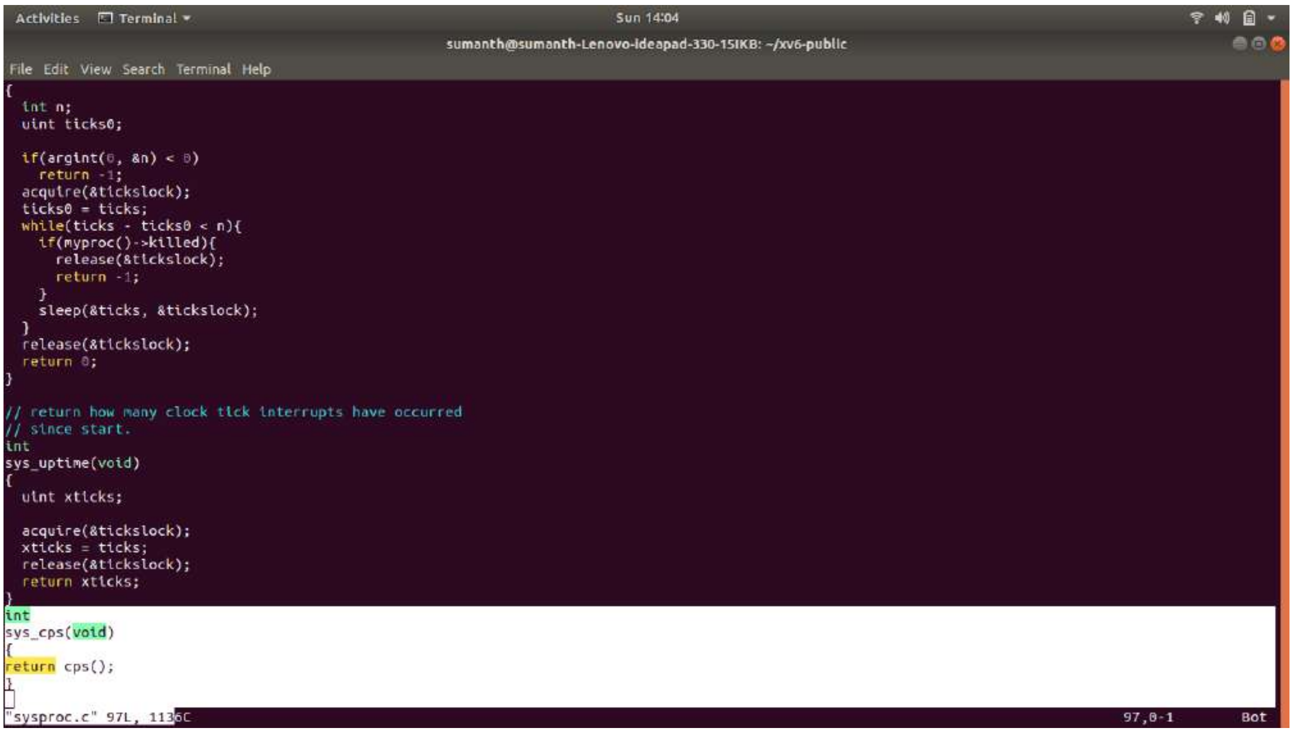

Activities
Terminal
Sun 14:04
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
// 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*);
int link(const char*, const char*);
int mkdir(const char*);
int chdir(const char*);
int dup(int);
int getpid(void);
char* sbrk(int);
int sleep(int);
int uptime(void);
int cps(void);

// ulib.c
int stat(const char*, struct stat*);
char* strcpy(char*, const char*);
void *memmove(void*, const void*, int);
char* strchr(const char*, char c);
int strcmp(const char*, const char*);
void printf(int, const char*, ...);
char* gets(char*, int max);
uint strlen(const char*);
void* memset(void*, int, uint);
void* malloc(uint);
void free(void*);
int atoi(const char*);
                                                                  40,1          Bot
```

We add this function prototype in syscalls

4. sysproc.c

We add the real implementation of our method here. We add a function sys_cps in the file sysproc.c which calls the function cps()

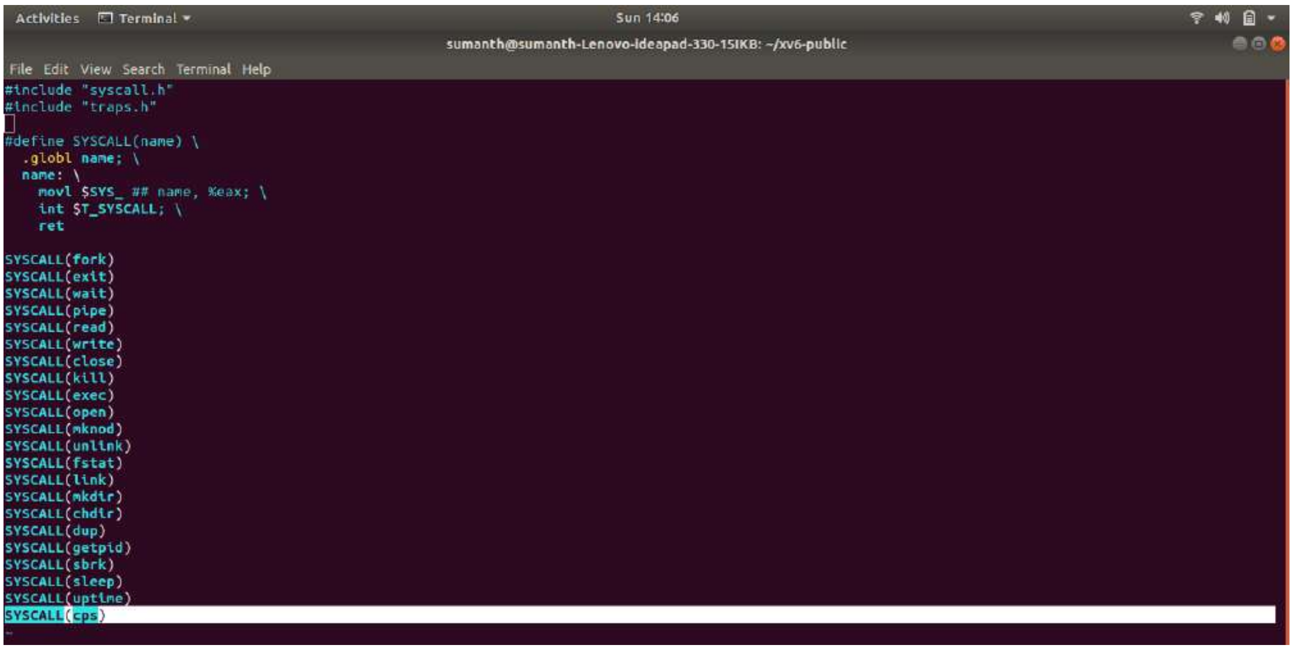


Activities
Terminal
Sun 14:04
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
{
  int n;
  uint ticks0;

  if(argint(0, &n) < 0)
    return -1;
  acquire(&tickslock);
  ticks0 = ticks;
  while(ticks - ticks0 < n){
    if(myproc()->killed){
      release(&tickslock);
      return -1;
    }
    sleep(&ticks, &tickslock);
  }
  release(&tickslock);
  return 0;
}

// return how many clock tick interrupts have occurred
// since start.
int
sys_uptime(void)
{
  uint xticks;

  acquire(&tickslock);
  xticks = ticks;
  release(&tickslock);
  return xticks;
}
int
sys_cps(void)
{
return cps();
}
"sysproc.c" 97L, 1136C                                    97,0-1        Bot
```

5. usys.s

It uses the macro to define connect the call of user to the system call function



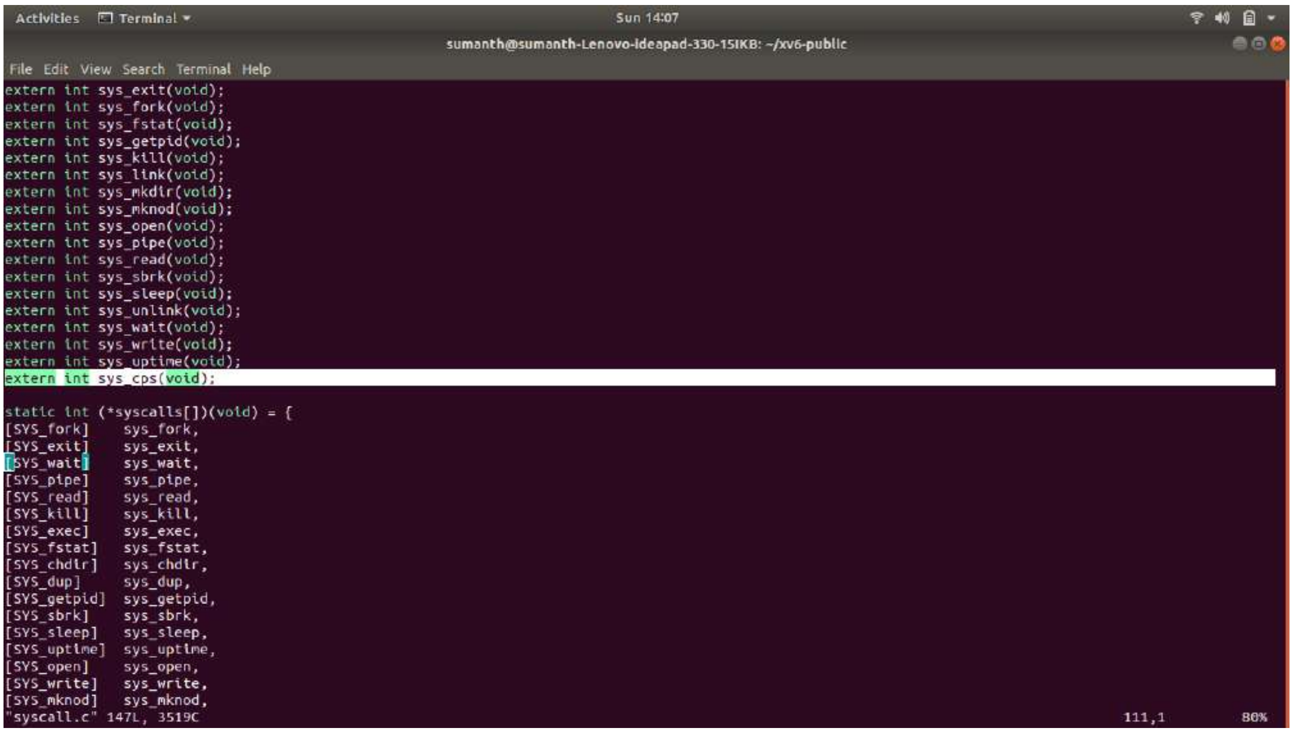
Activities
Terminal
Sun 14:06
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
#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)
SYSCALL(exec)
SYSCALL(open)
SYSCALL(mknod)
SYSCALL(unlink)
SYSCALL(fstat)
SYSCALL(link)
SYSCALL(mkdir)
SYSCALL(chdir)
SYSCALL(dup)
SYSCALL(getpid)
SYSCALL(sbrk)
SYSCALL(sleep)
SYSCALL(uptime)
SYSCALL(cps)
~
```

Then we add this to usys.s

6. syscall.c

It defines the function that connects the kernel and the shell and by using the position defined in syscall.h it adds the function to the system call


Activities
Terminal
Sun 14:07
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
extern int sys_exit(void);
extern int sys_fork(void);
extern int sys_fstat(void);
extern int sys_getpid(void);
extern int sys_kill(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);
extern int sys_cps(void);

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,
[SYS_mknod]   sys_mknod,
"syscall.c" 147L, 3519C                                        111,1          80%
```

7. proc.c

We add this code to proc.c

Explanation of code

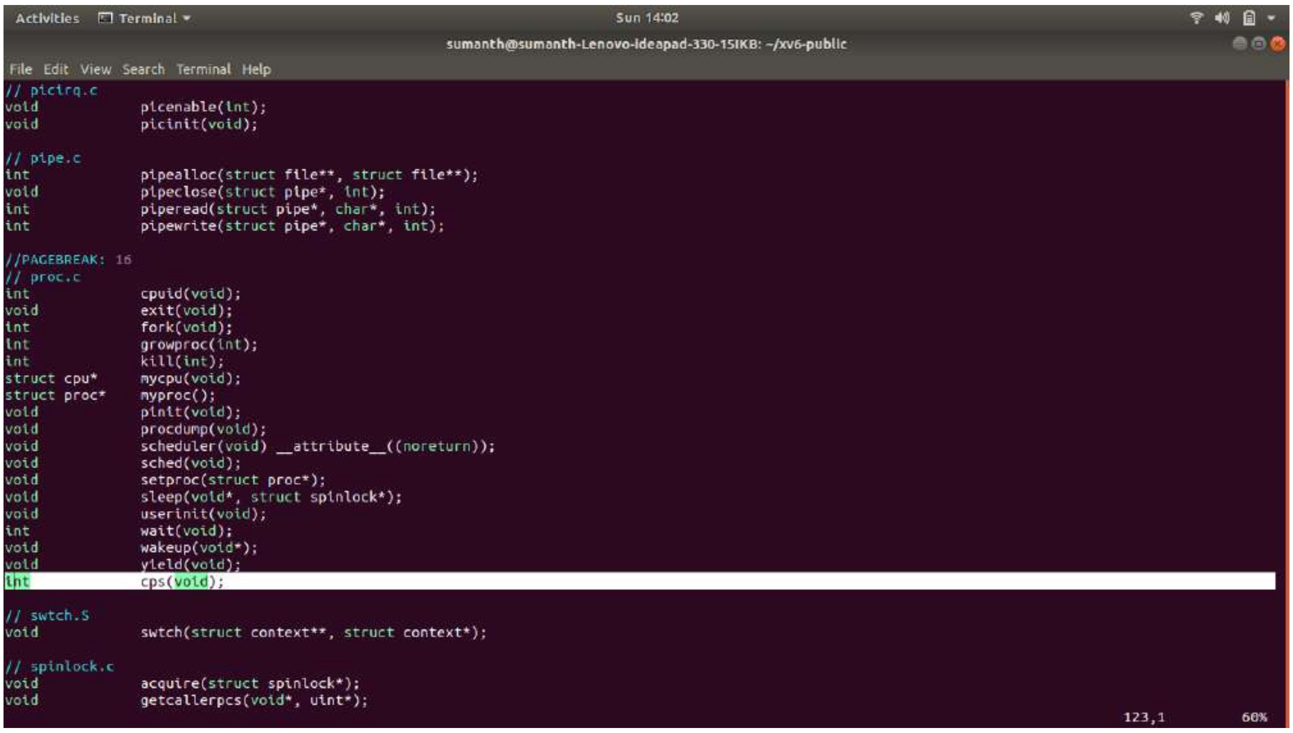
• It interrupts on the processor

• It acquires a lock

• It runs through the process table and checks whether the process is SLEEPING or RUNNING or RUNNABLE and then prints the same pid and status of the process

• It releases the lock

• It returns the syscall number which is 22



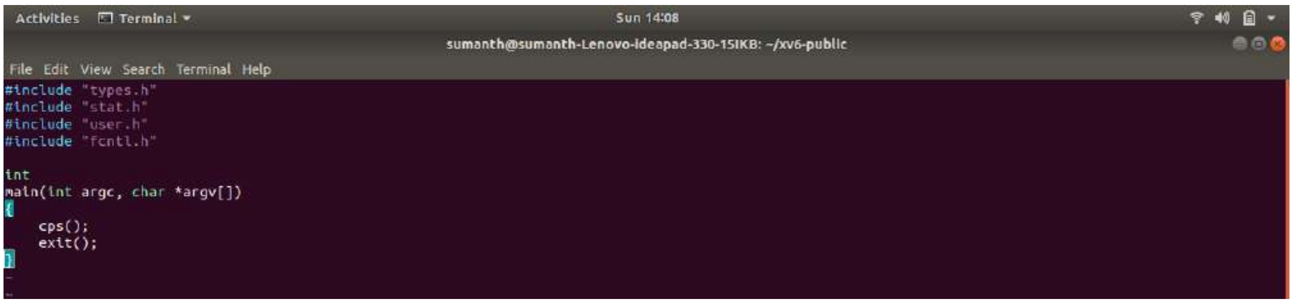
Activities   Terminal ▾   Sun 14:02

sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public

File Edit View Search Terminal Help

```
// picirq.c
void            picenable(int);
void            picinit(void);

// pipe.c
int             pipealloc(struct file**, struct file**);
void            pipeclose(struct pipe*, int);
int             piperead(struct pipe*, char*, int);
int             pipewrite(struct pipe*, char*, int);

//PAGEBREAK: 16
// proc.c
int             cpuid(void);
void            exit(void);
int             fork(void);
int             growproc(int);
int             kill(int);
struct cpu*     mycpu(void);
struct proc*    myproc();
void            pinit(void);
void            procdump(void);
void            scheduler(void) __attribute__((noreturn));
void            sched(void);
void            setproc(struct proc*);
void            sleep(void*, struct spinlock*);
void            userinit(void);
int             wait(void);
void            wakeup(void*);
void            yield(void);
int             cps(void);

// swtch.S
void            swtch(struct context**, struct context*);

// spinlock.c
void            acquire(struct spinlock*);
void            getcallerpcs(void*, uint*);
                                                                123,1          60%
```

8. ps.c



Activities   Terminal ▾   Sun 14:08

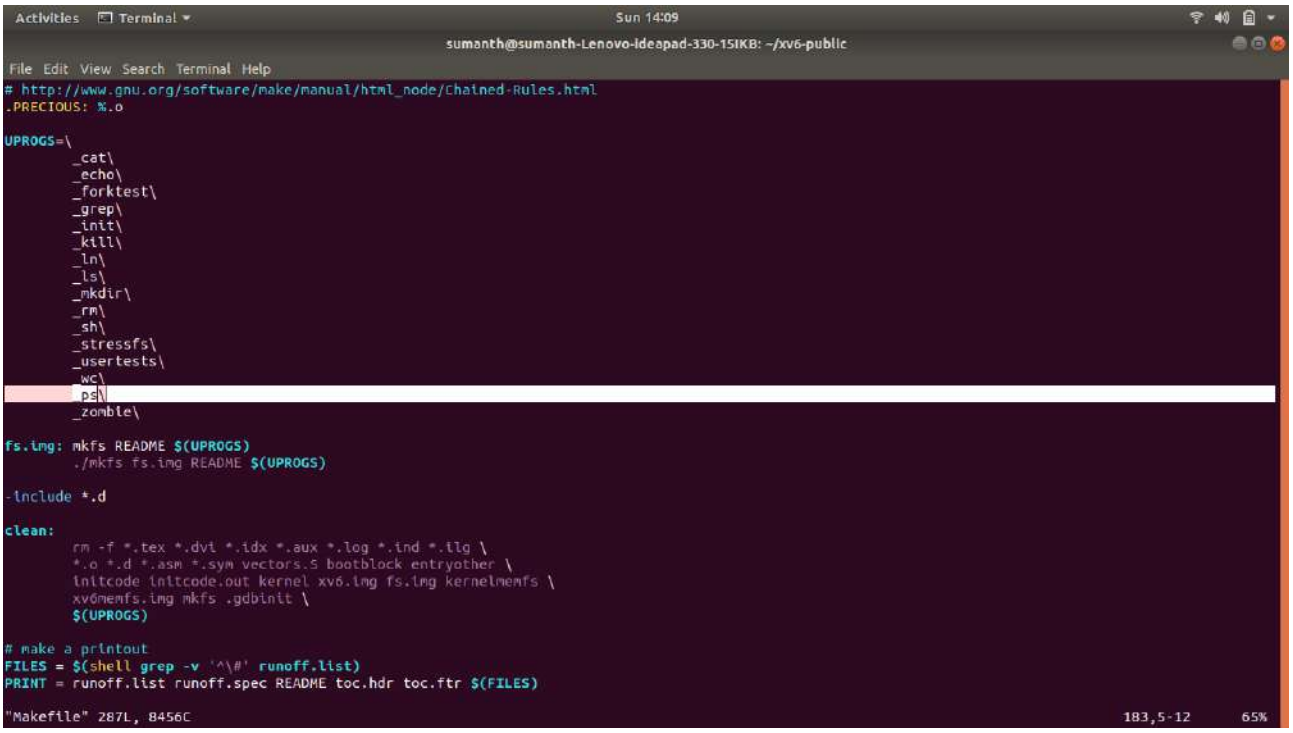
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public

File Edit View Search Terminal Help

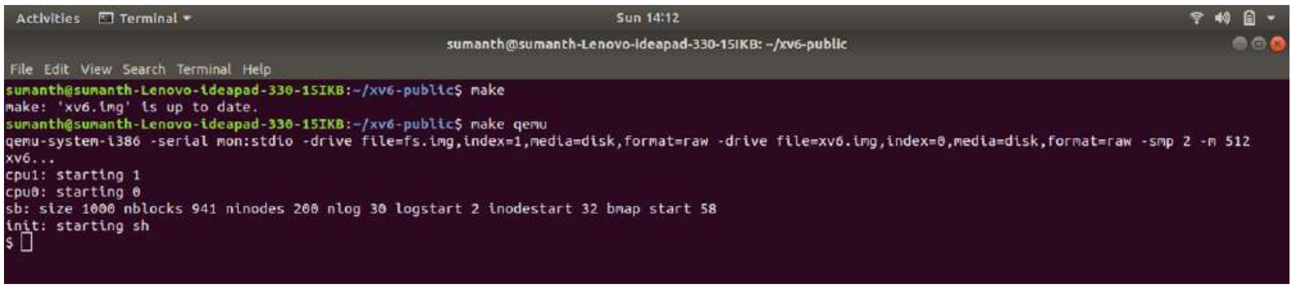
```
#include "types.h"
#include "stat.h"
#include "user.h"
#include "fcntl.h"

int
main(int argc, char *argv[])
{
    cps();
    exit();
}
~
~
```

9.Makefile

Then we make which compiles all changes we made inside the xv6 directories and subdirectories .Our new syscall is now visible in the list


Activities
Terminal
Sun 14:09
sumanth@sumanth-Lenovo-Ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

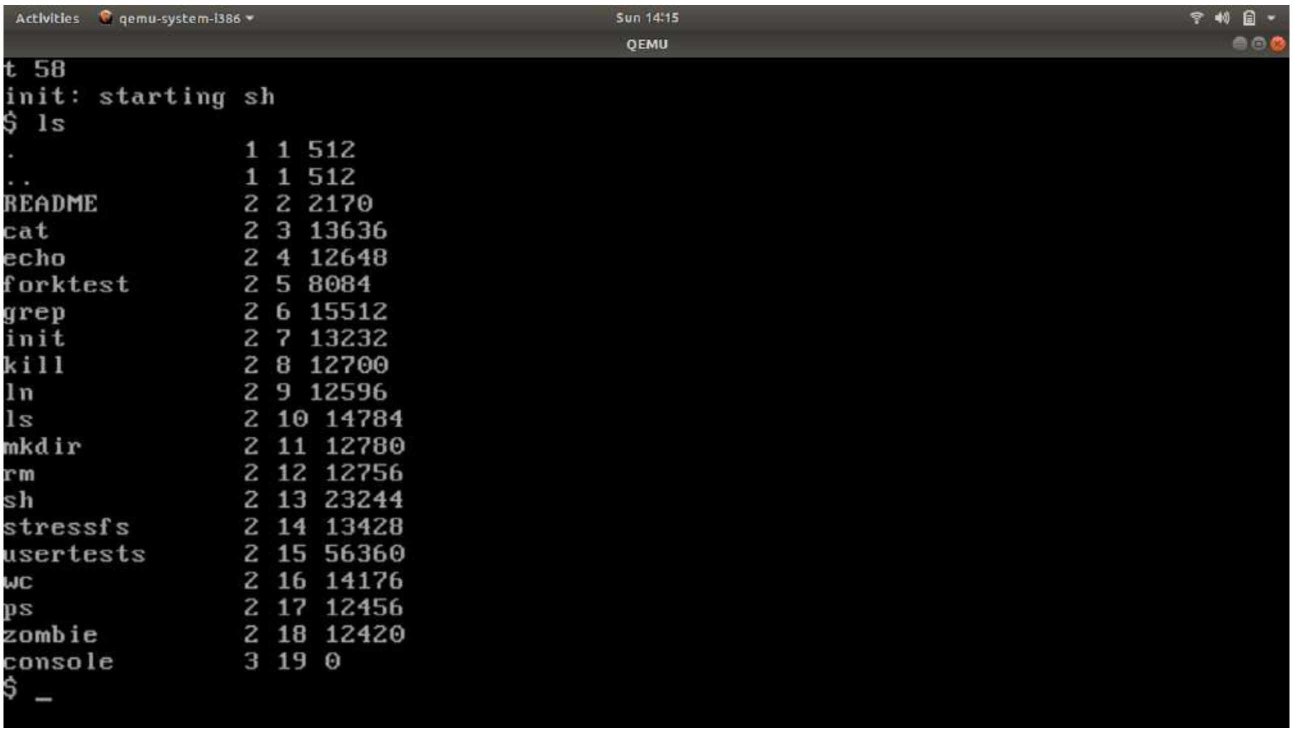
```
# http://www.gnu.org/software/make/manual/html_node/Chained-Rules.html
.PRECIOUS: %.o

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

fs.img: mkfs README $(UPROGS)
	./mkfs fs.img README $(UPROGS)

-include *.d

clean:
	rm -f *.tex *.dvi *.idx *.aux *.log *.ind *.ilg \
	*.o *.d *.asm *.sym vectors.S bootblock entryother \
	initcode initcode.out kernel xv6.img fs.img kernelmemfs \
	xv6memfs.img mkfs .gdbinit \
	$(UPROGS)

# make a printout
FILES = $(shell grep -v '^\#' runoff.list)
PRINT = runoff.list runoff.spec README toc.hdr toc.ftr $(FILES)

"Makefile" 287L, 8456C                                        183,5-12      65%
```


Activities
Terminal
Sun 14:12
sumanth@sumanth-Lenovo-Ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

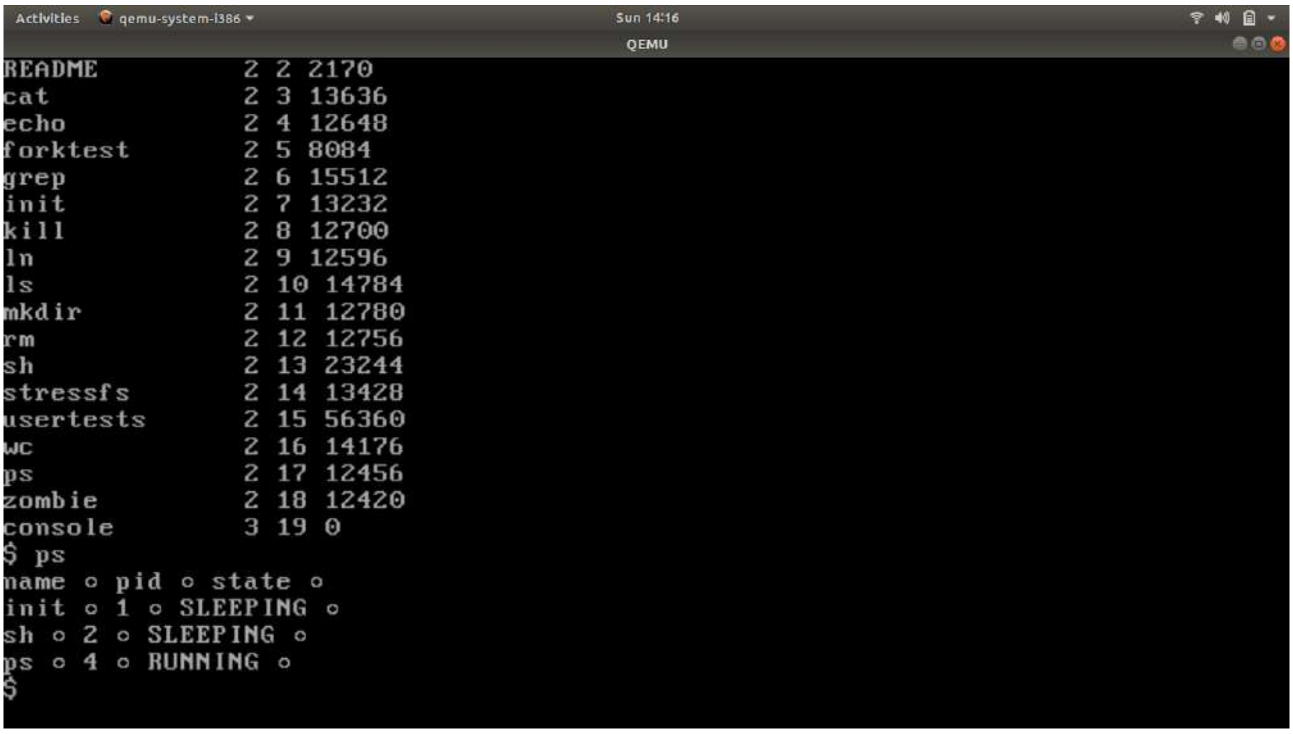
```
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ make
make: 'xv6.img' is up to date.
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ make qemu
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ 
```

Now we will compile the whole code and execute the os according to the changes made using make command

Our new syscall is now visible in the list:



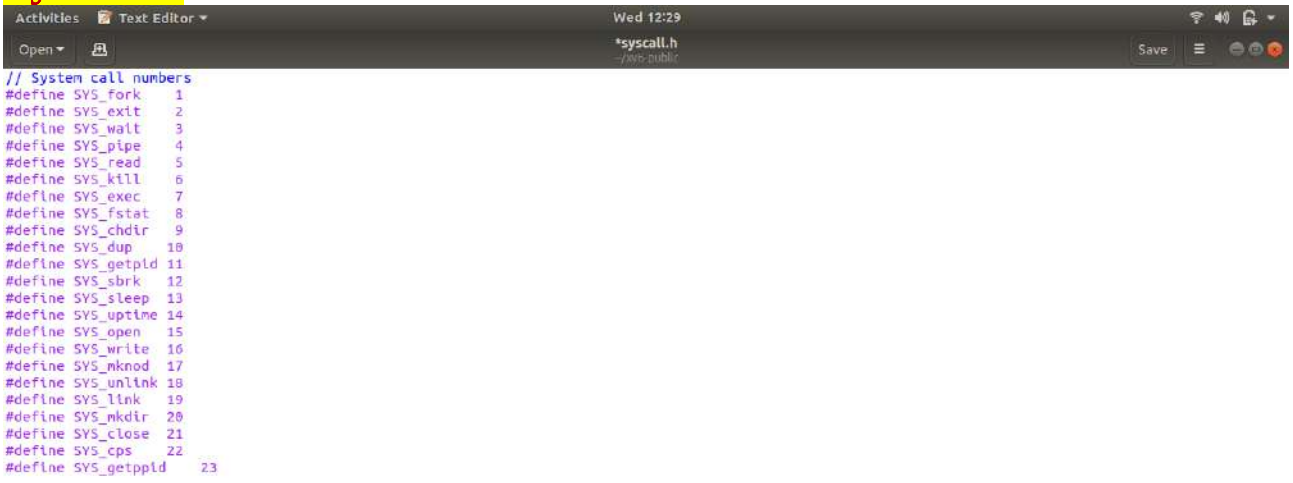
```
t 58
init: starting sh
$ ls
.              1 1 512
..             1 1 512
README         2 2 2170
cat            2 3 13636
echo           2 4 12648
forktest       2 5 8084
grep           2 6 15512
init           2 7 13232
kill           2 8 12700
ln             2 9 12596
ls             2 10 14784
mkdir          2 11 12780
rm             2 12 12756
sh             2 13 23244
stressfs       2 14 13428
usertests      2 15 56360
wc             2 16 14176
ps             2 17 12456
zombie         2 18 12420
console        3 19 0
$ _
```



```
README         2 2 2170
cat            2 3 13636
echo           2 4 12648
forktest       2 5 8084
grep           2 6 15512
init           2 7 13232
kill           2 8 12700
ln             2 9 12596
ls             2 10 14784
mkdir          2 11 12780
rm             2 12 12756
sh             2 13 23244
stressfs       2 14 13428
usertests      2 15 56360
wc             2 16 14176
ps             2 17 12456
zombie         2 18 12420
console        3 19 0
$ ps
name o pid o state o
init o 1 o SLEEPING o
sh o 2 o SLEEPING o
ps o 4 o RUNNING o
$
```

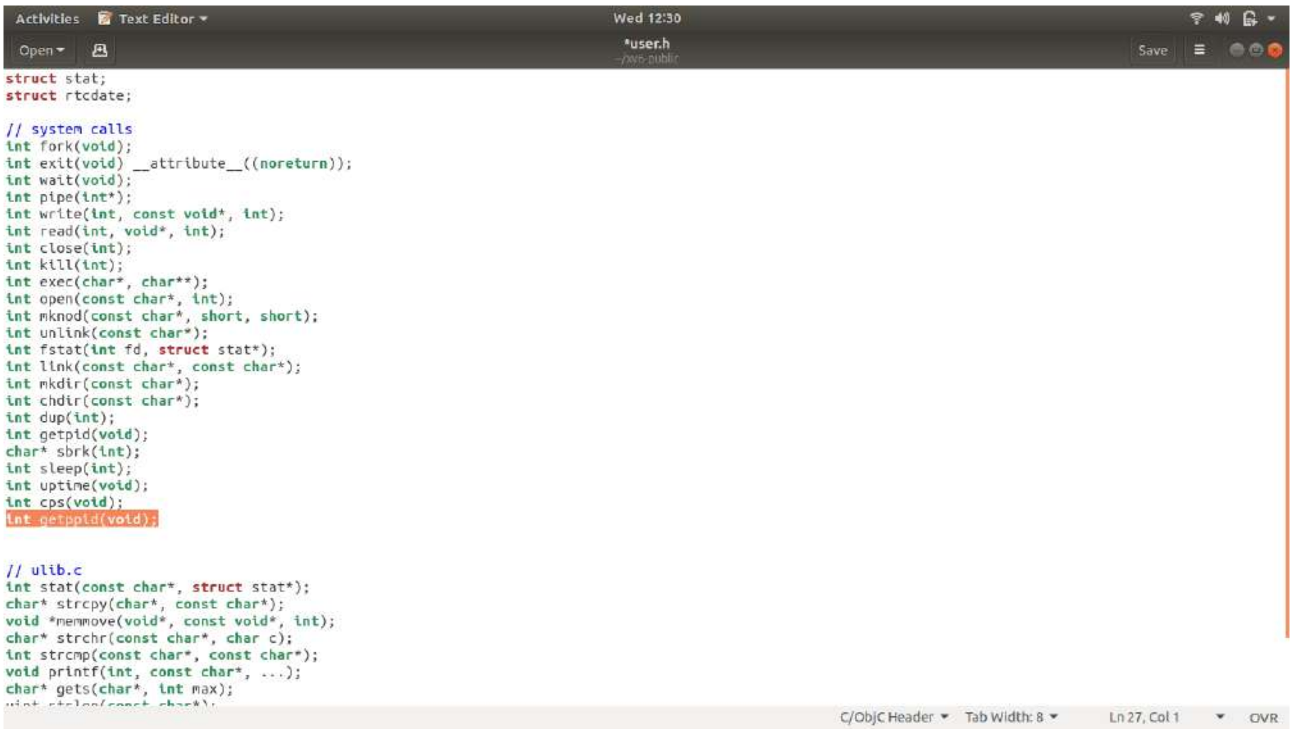
Creating syscall Parent

Syscall.h

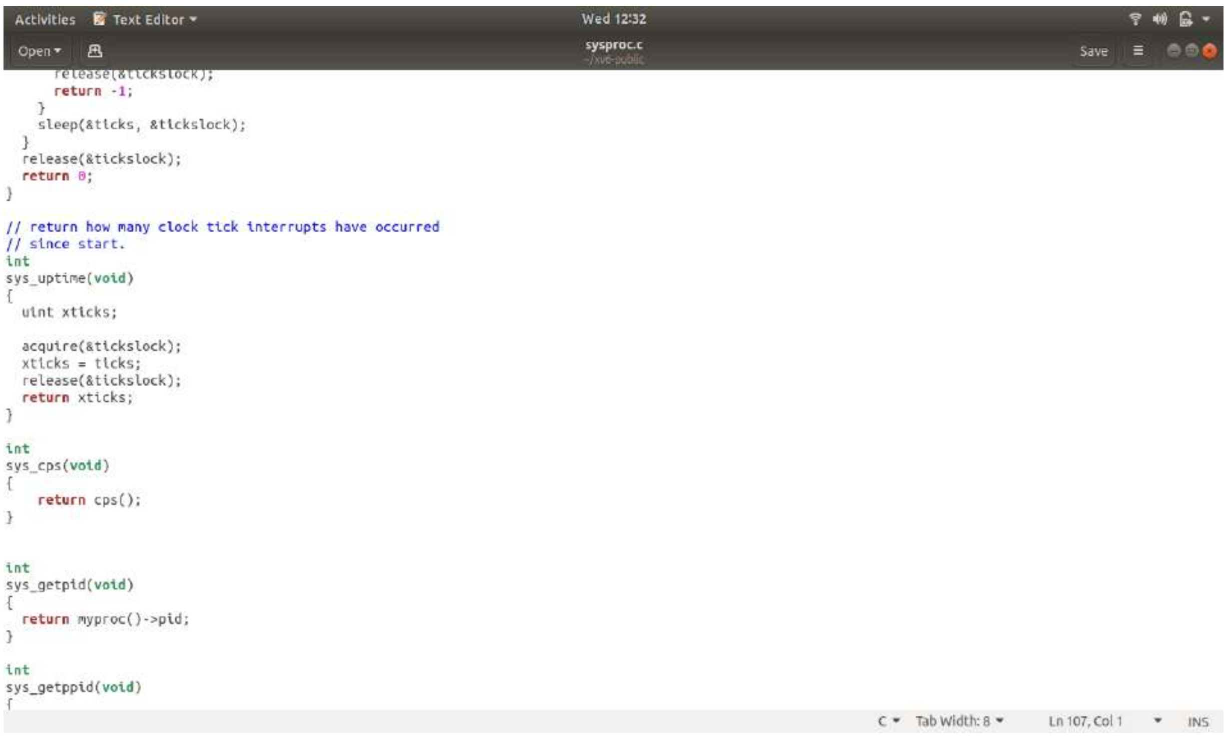


```c
// System call numbers
#define SYS_fork    1
#define SYS_exit    2
#define SYS_wait    3
#define SYS_pipe    4
#define SYS_read    5
#define SYS_kill    6
#define SYS_exec    7
#define SYS_fstat   8
#define SYS_chdir   9
#define SYS_dup    10
#define SYS_getpid 11
#define SYS_sbrk   12
#define SYS_sleep  13
#define SYS_uptime 14
#define SYS_open   15
#define SYS_write  16
#define SYS_mknod  17
#define SYS_unlink 18
#define SYS_link   19
#define SYS_mkdir  20
#define SYS_close  21
#define SYS_cps    22
#define SYS_getppid     23
```

User.h

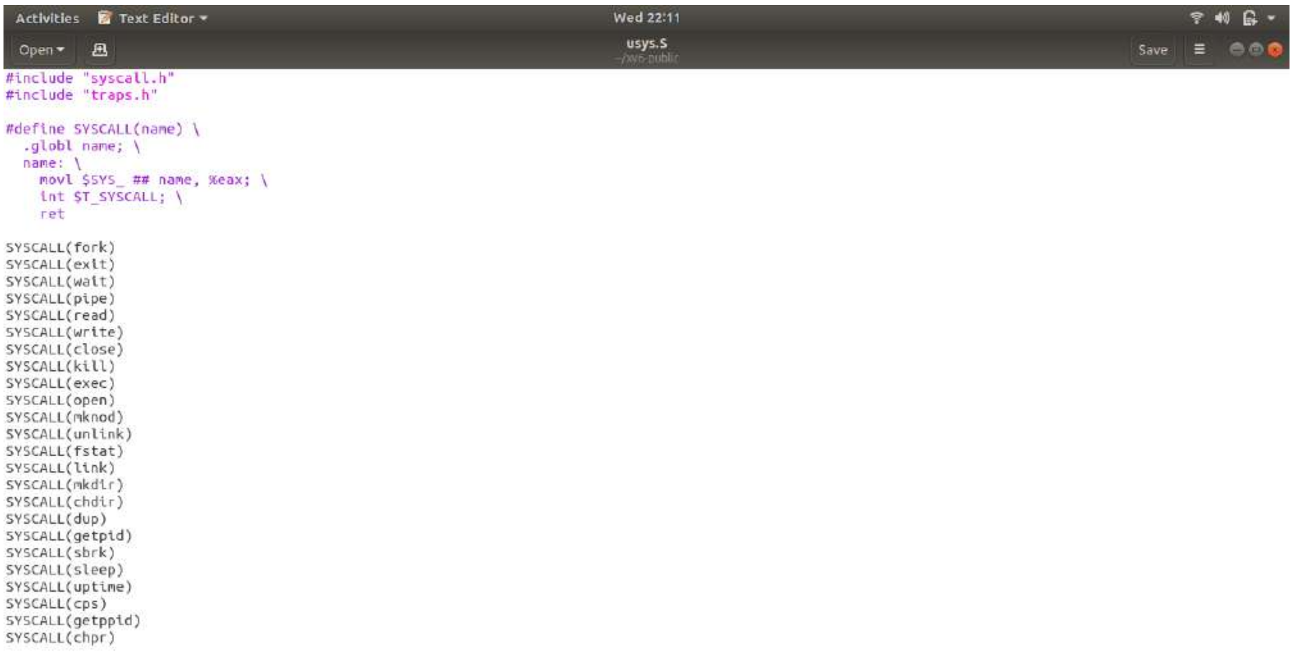


```c
struct stat;
struct rtcdate;

// 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*);
int link(const char*, const char*);
int mkdir(const char*);
int chdir(const char*);
int dup(int);
int getpid(void);
char* sbrk(int);
int sleep(int);
int uptime(void);
int cps(void);
int getppid(void);


// ulib.c
int stat(const char*, struct stat*);
char* strcpy(char*, const char*);
void *memmove(void*, const void*, int);
char* strchr(const char*, char c);
int strcmp(const char*, const char*);
void printf(int, const char*, ...);
char* gets(char*, int max);
```



```
      release(&tickslock);
      return -1;
    }
    sleep(&ticks, &tickslock);
  }
  release(&tickslock);
  return 0;
}

// return how many clock tick interrupts have occurred
// since start.
int
sys_uptime(void)
{
  uint xticks;

  acquire(&tickslock);
  xticks = ticks;
  release(&tickslock);
  return xticks;
}

int
sys_cps(void)
{
    return cps();
}


int
sys_getpid(void)
{
  return myproc()->pid;
}

int
sys_getppid(void)
{
```

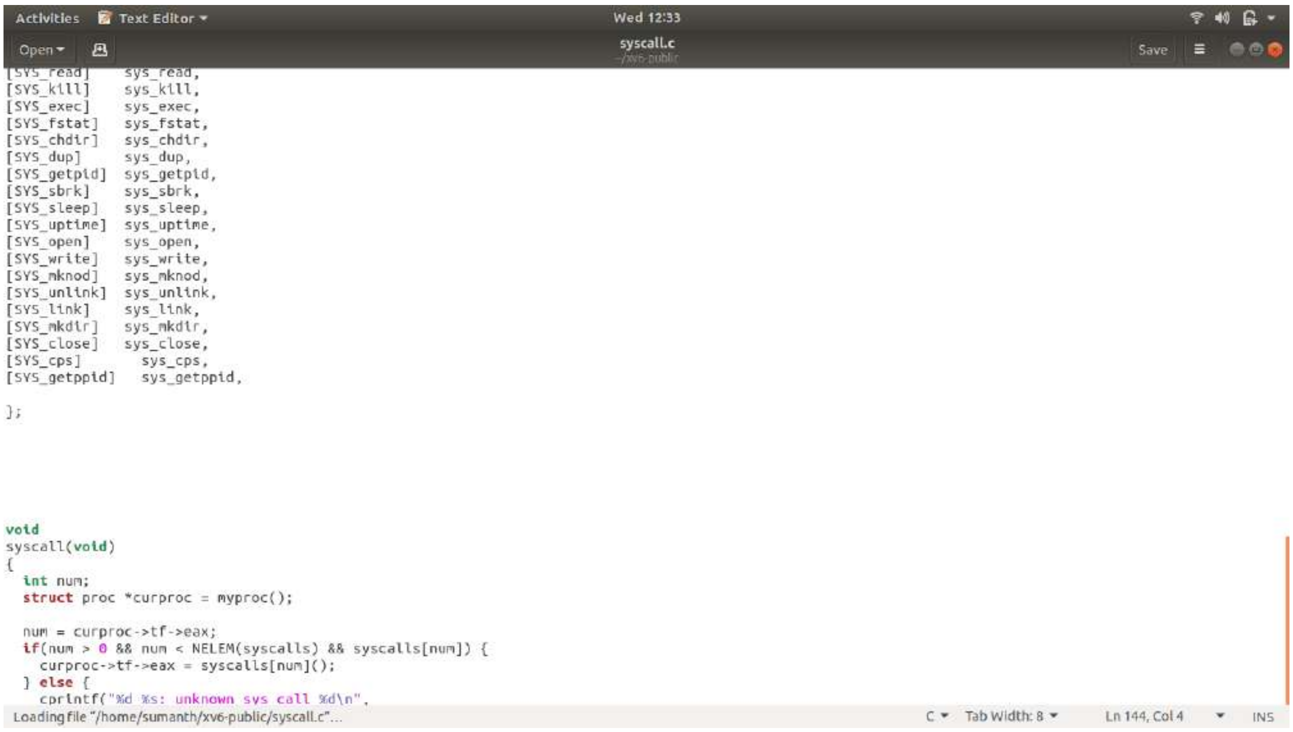
C ▾ Tab Width: 8 ▾ Ln 107, Col 1 ▾ INS



```
#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)
SYSCALL(exec)
SYSCALL(open)
SYSCALL(mknod)
SYSCALL(unlink)
SYSCALL(fstat)
SYSCALL(link)
SYSCALL(mkdir)
SYSCALL(chdir)
SYSCALL(dup)
SYSCALL(getpid)
SYSCALL(sbrk)
SYSCALL(sleep)
SYSCALL(uptime)
SYSCALL(cps)
SYSCALL(getppid)
SYSCALL(chpr)
```

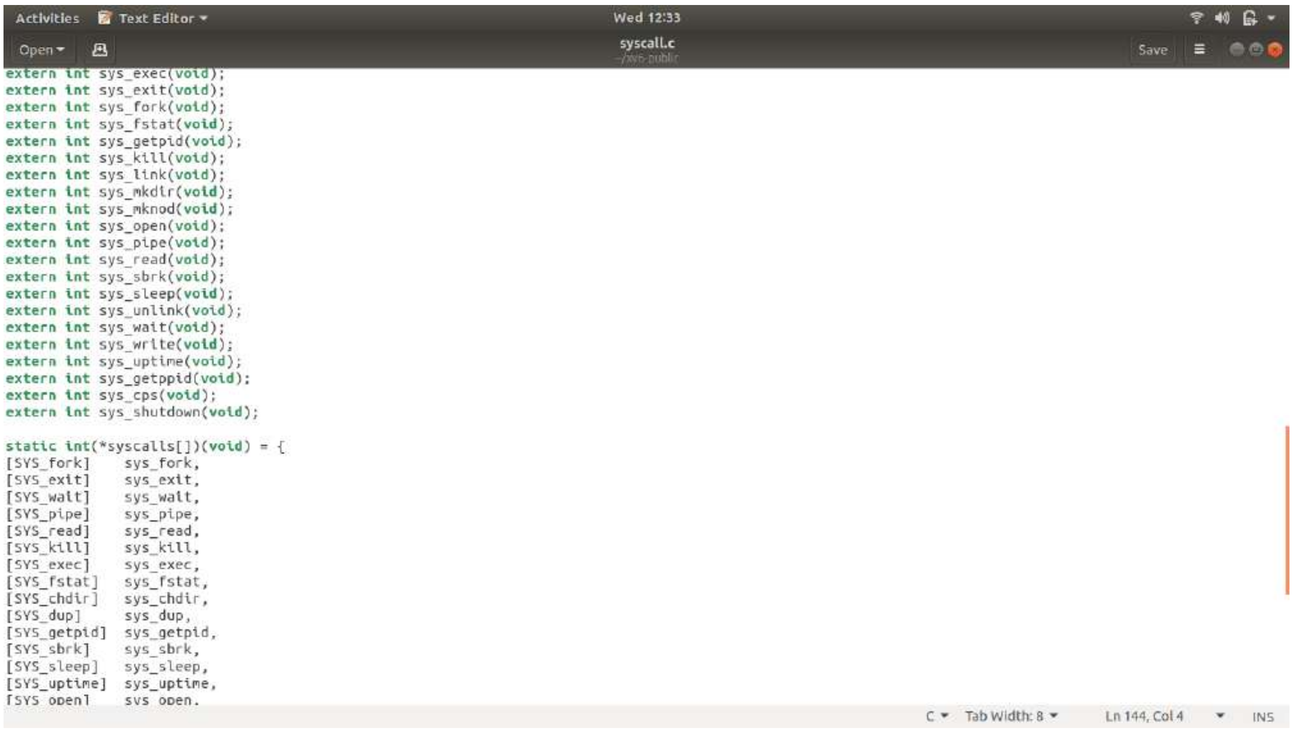
usys.S

Syscall.c



```
[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,
[SYS_mknod]   sys_mknod,
[SYS_unlink]  sys_unlink,
[SYS_link]    sys_link,
[SYS_mkdir]   sys_mkdir,
[SYS_close]   sys_close,
[SYS_cps]       sys_cps,
[SYS_getppid]   sys_getppid,

};




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",
```

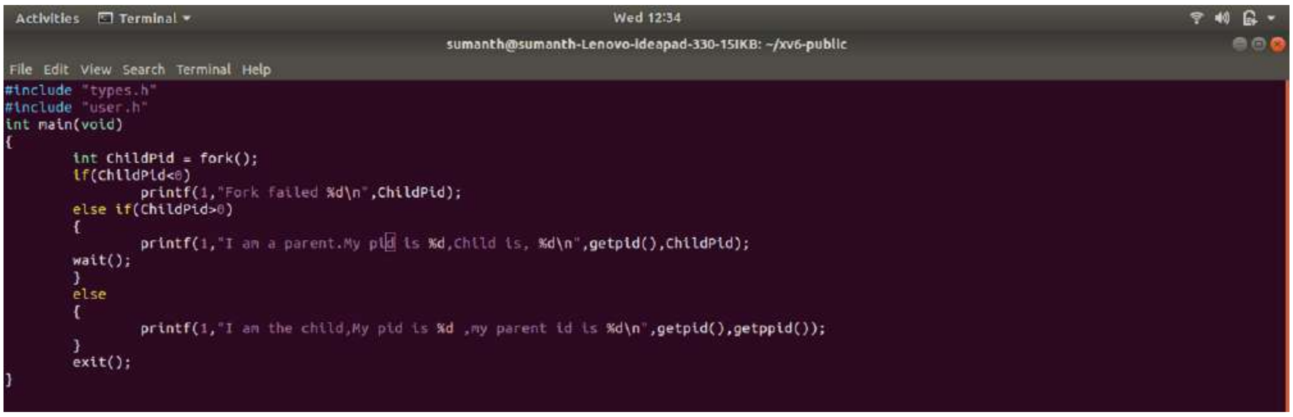
```
Loading file "/home/sumanth/xv6-public/syscall.c"...        C ▾   Tab Width: 8 ▾   Ln 144, Col 4   ▾   INS
```



```
extern int sys_exec(void);
extern int sys_exit(void);
extern int sys_fork(void);
extern int sys_fstat(void);
extern int sys_getpid(void);
extern int sys_kill(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);
extern int sys_getppid(void);
extern int sys_cps(void);
extern int sys_shutdown(void);

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,
```

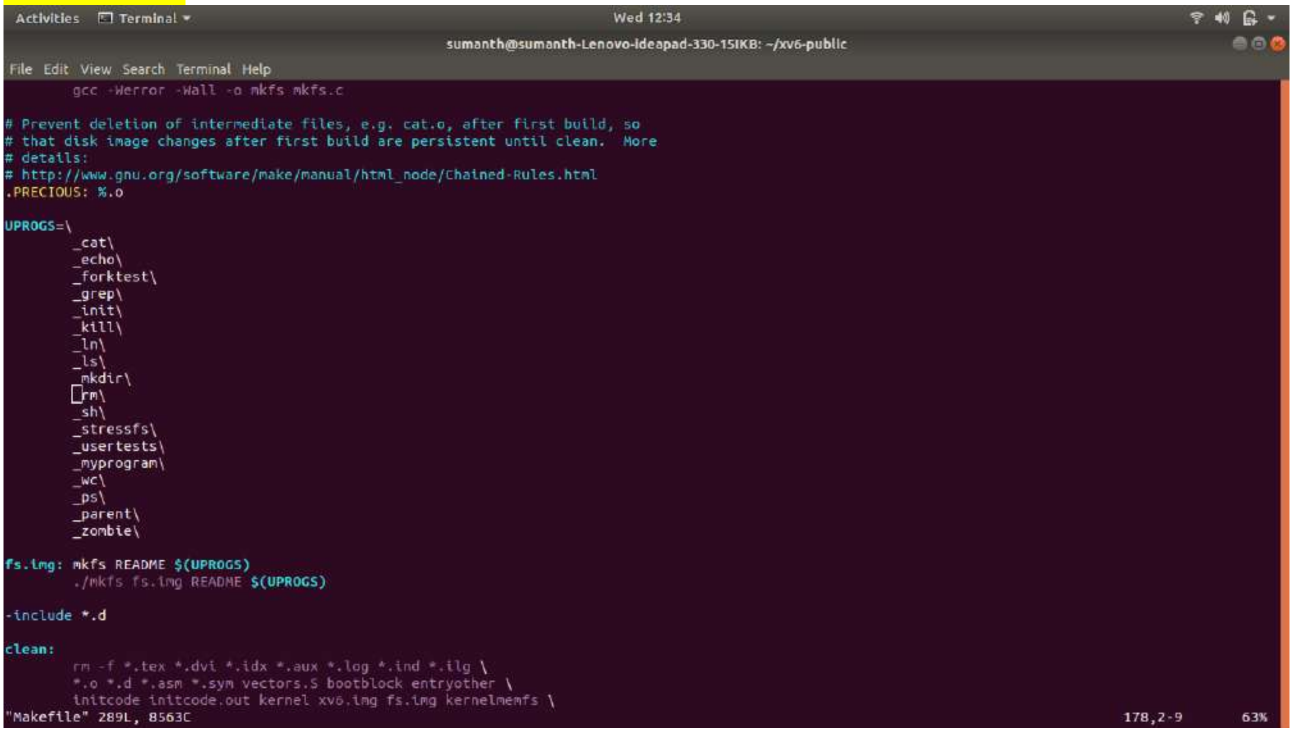
```
C ▾   Tab Width: 8 ▾   Ln 144, Col 4   ▾   INS
```

Parent.c

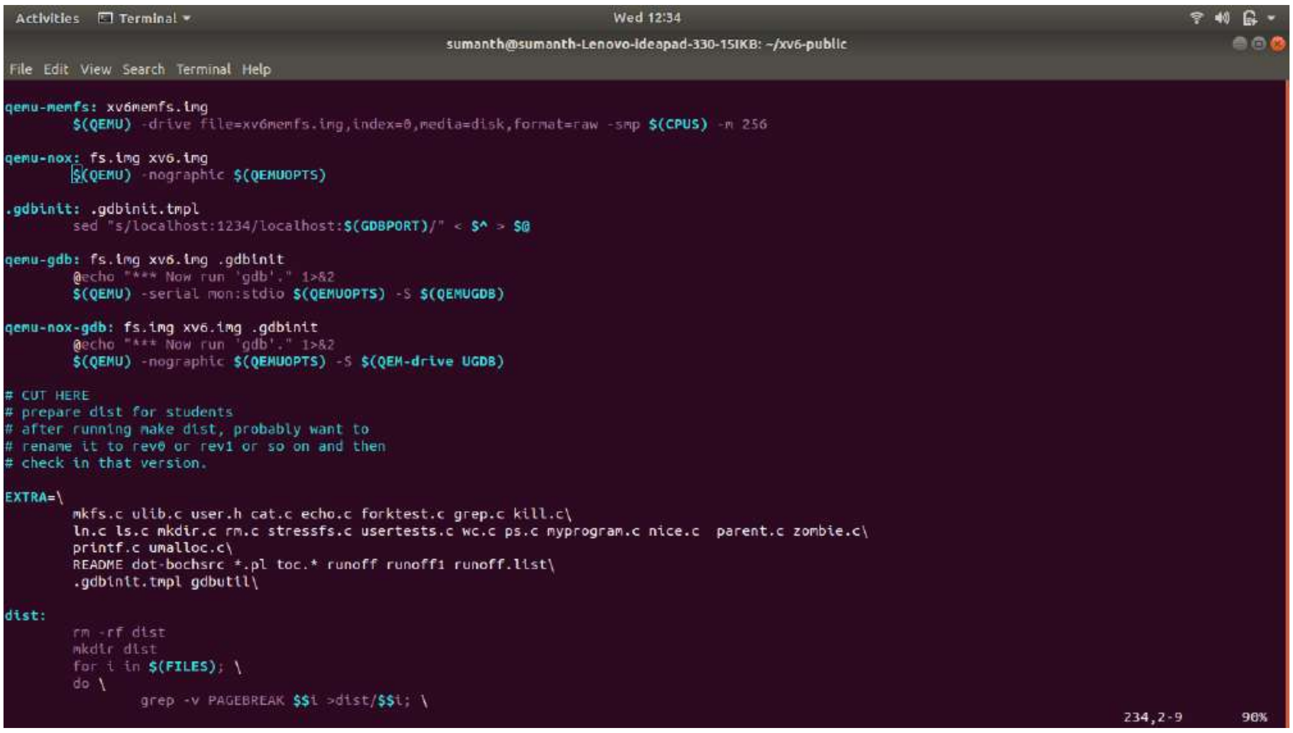

Activities
Terminal
Wed 12:34
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
#include "types.h"
#include "user.h"
int main(void)
{
        int ChildPid = fork();
        if(ChildPid<0)
                printf(1,"Fork failed %d\n",ChildPid);
        else if(ChildPid>0)
        {
                printf(1,"I am a parent.My pid is %d,Child is, %d\n",getpid(),ChildPid);
        wait();
        }
        else
        {
                printf(1,"I am the child,My pid is %d ,my parent id is %d\n",getpid(),getppid());
        }
        exit();
}
```

Makefile

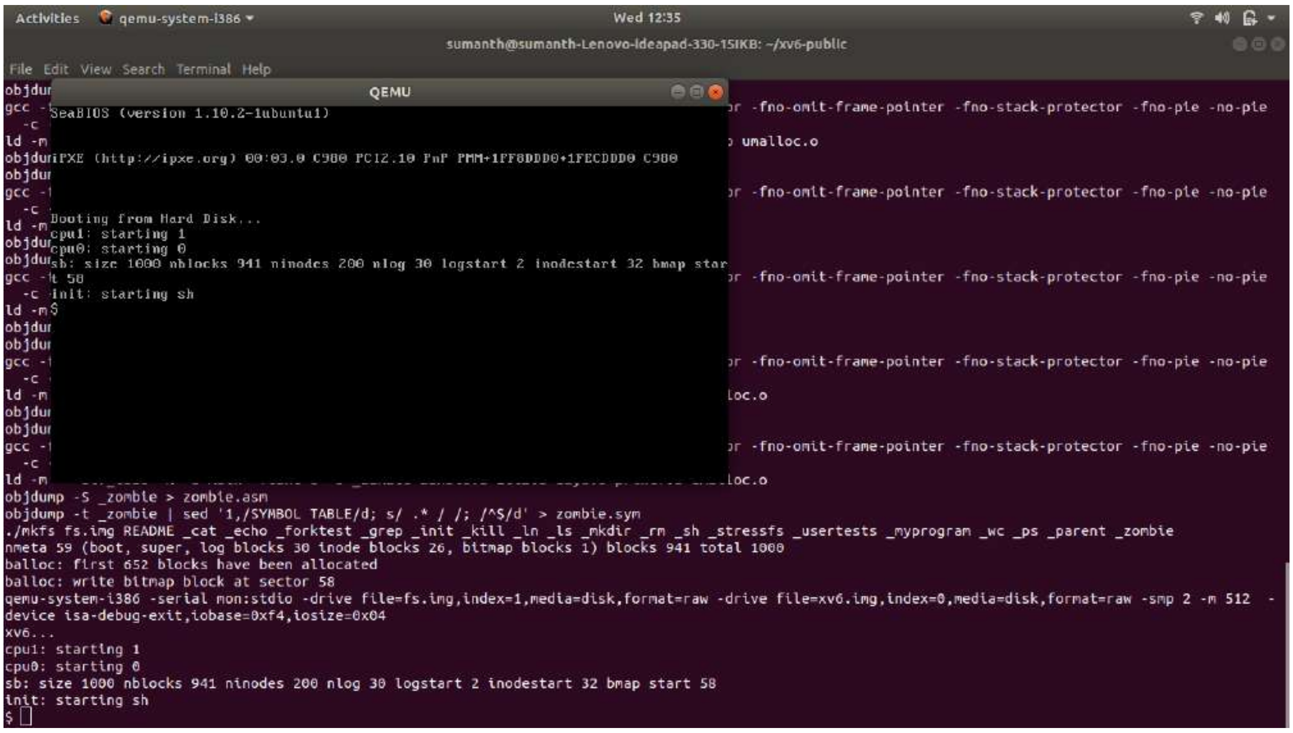

Activities
Terminal
Wed 12:34
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
        gcc -Werror -Wall -o mkfs mkfs.c

# Prevent deletion of intermediate files, e.g. cat.o, after first build, so
# that disk image changes after first build are persistent until clean.  More
# details:
# http://www.gnu.org/software/make/manual/html_node/Chained-Rules.html
.PRECIOUS: %.o

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

fs.img: mkfs README $(UPROGS)
        ./mkfs fs.img README $(UPROGS)

-include *.d

clean:
        rm -f *.tex *.dvi *.idx *.aux *.log *.ind *.ilg \
        *.o *.d *.asm *.sym vectors.S bootblock entryother \
        initcode initcode.out kernel xv6.img fs.img kernelmemfs \
"Makefile" 289L, 8563C                                                        178,2-9        63%
```


Activities
Terminal
Wed 12:34
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

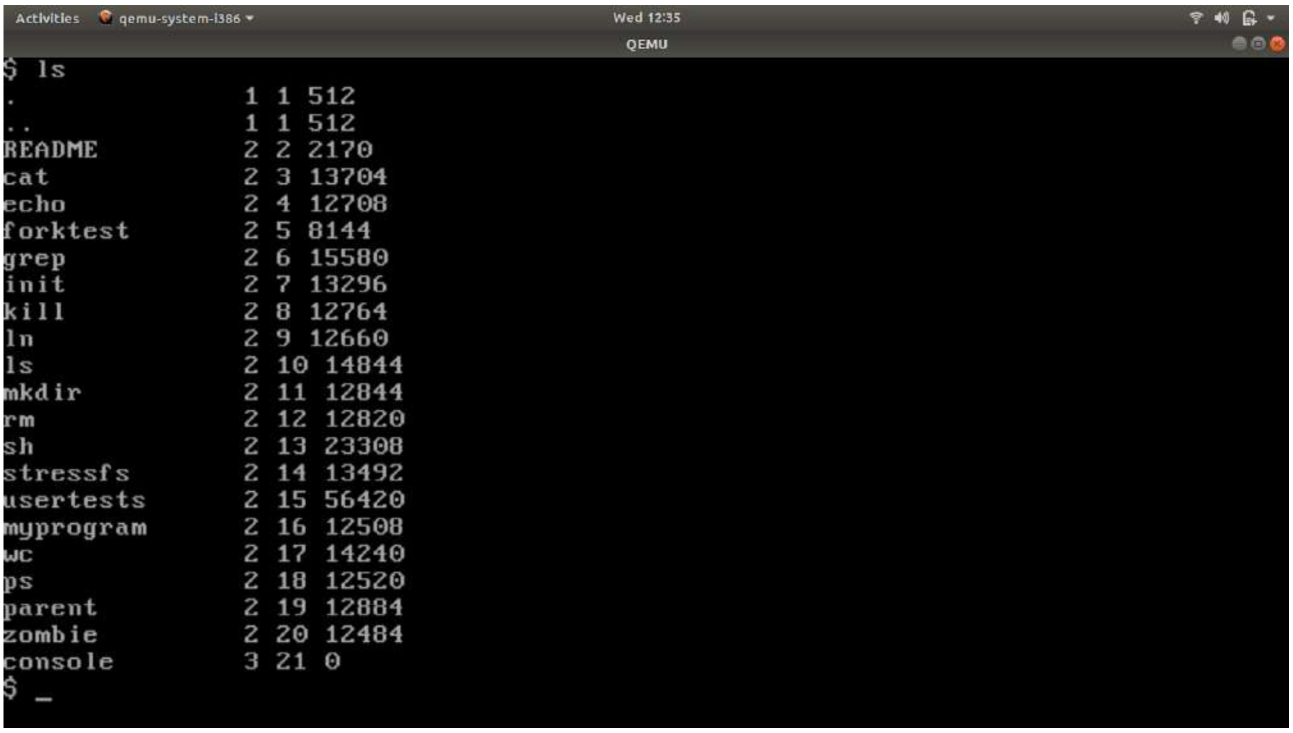
```
qemu-memfs: xv6memfs.img
        $(QEMU) -drive file=xv6memfs.img,index=0,media=disk,format=raw -smp $(CPUS) -m 256

qemu-nox: fs.img xv6.img
        $(QEMU) -nographic $(QEMUOPTS)

.gdbinit: .gdbinit.tmpl
        sed "s/localhost:1234/localhost:$(GDBPORT)/" < $^ > $@

qemu-gdb: fs.img xv6.img .gdbinit
        @echo "*** Now run 'gdb'." 1>&2
        $(QEMU) -serial mon:stdio $(QEMUOPTS) -S $(QEMUGDB)

qemu-nox-gdb: fs.img xv6.img .gdbinit
        @echo "*** Now run 'gdb'." 1>&2
        $(QEMU) -nographic $(QEMUOPTS) -S $(QEM-drive UGDB)

# CUT HERE
# prepare dist for students
# after running make dist, probably want to
# rename it to rev0 or rev1 or so on and then
# check in that version.

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 ps.c myprogram.c nice.c  parent.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 \
                grep -v PAGEBREAK $$i >dist/$$i; \
                                                                                  234,2-9       90%
```

Customized system call parent is now visible on qemu editor

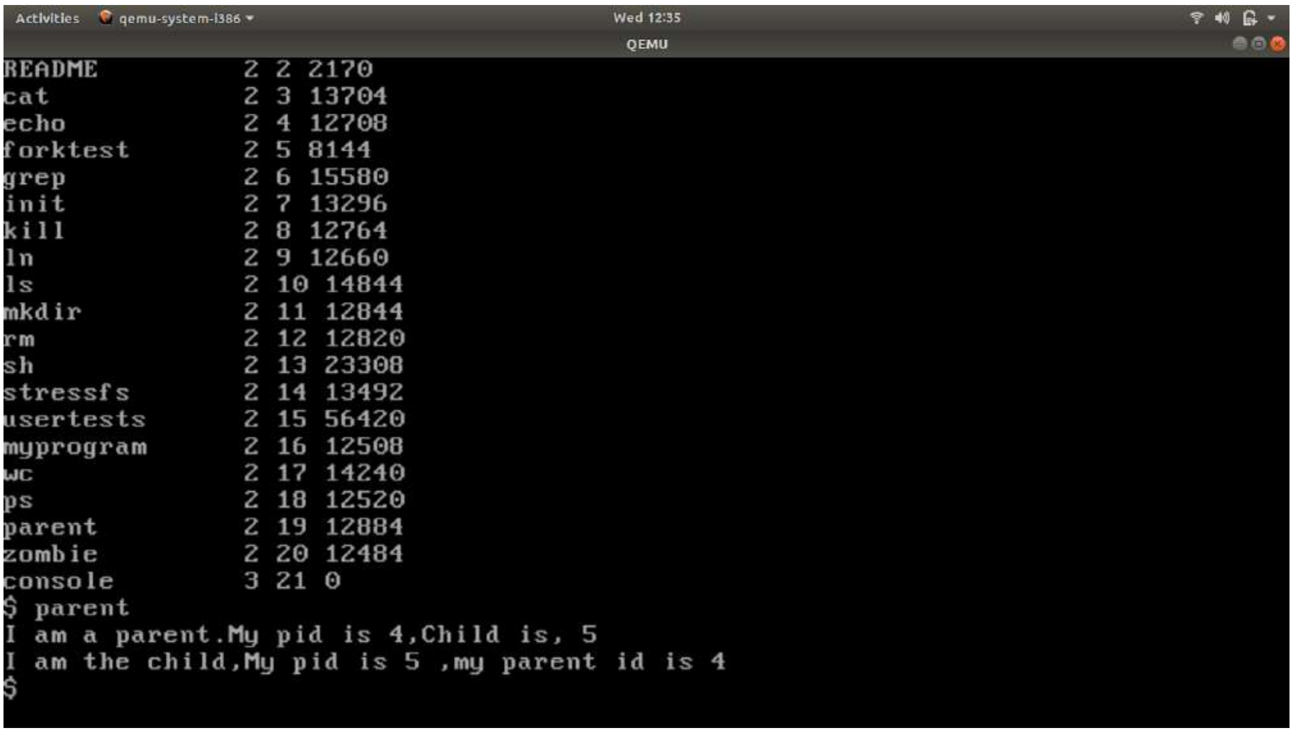

Activities
qemu-system-i386
Wed 12:35
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help
QEMU

```
SeaBIOS (version 1.10.2-1ubuntu1)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF8DDD0+1FECDDD0 C980


Booting from Hard Disk...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$
```

```
objdu
gcc -                                 or -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c
ld -m                                 umalloc.o
objdu
objdu
gcc -                                 or -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c
ld -m
objdu
objdu
gcc -                                 or -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c
ld -m
objdu
objdu
gcc -                                 or -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c
ld -m                                 loc.o
objdu
objdu
gcc -                                 or -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c
ld -m                                 loc.o
objdump -S _zombie > zombie.asm
objdump -t _zombie | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > zombie.sym
./mkfs fs.img README _cat _echo _forktest _grep _init _kill _ln _ls _mkdir _rm _sh _stressfs _usertests _myprogram _wc _ps _parent _zombie
nmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks 941 total 1000
balloc: first 652 blocks have been allocated
balloc: write bitmap block at sector 58
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512  -
device isa-debug-exit,iobase=0xf4,iosize=0x04
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ 
```



```
$ ls
.              1 1 512
..             1 1 512
README         2 2 2170
cat            2 3 13704
echo           2 4 12708
forktest       2 5 8144
grep           2 6 15580
init           2 7 13296
kill           2 8 12764
ln             2 9 12660
ls             2 10 14844
mkdir          2 11 12844
rm             2 12 12820
sh             2 13 23308
stressfs       2 14 13492
usertests      2 15 56420
myprogram      2 16 12508
wc             2 17 14240
ps             2 18 12520
parent         2 19 12884
zombie         2 20 12484
console        3 21 0
$ _
```



```
README         2 2 2170
cat            2 3 13704
echo           2 4 12708
forktest       2 5 8144
grep           2 6 15580
init           2 7 13296
kill           2 8 12764
ln             2 9 12660
ls             2 10 14844
mkdir          2 11 12844
rm             2 12 12820
sh             2 13 23308
stressfs       2 14 13492
usertests      2 15 56420
myprogram      2 16 12508
wc             2 17 14240
ps             2 18 12520
parent         2 19 12884
zombie         2 20 12484
console        3 21 0
$ parent
I am a parent.My pid is 4,Child is, 5
I am the child,My pid is 5 ,my parent id is 4
$
```

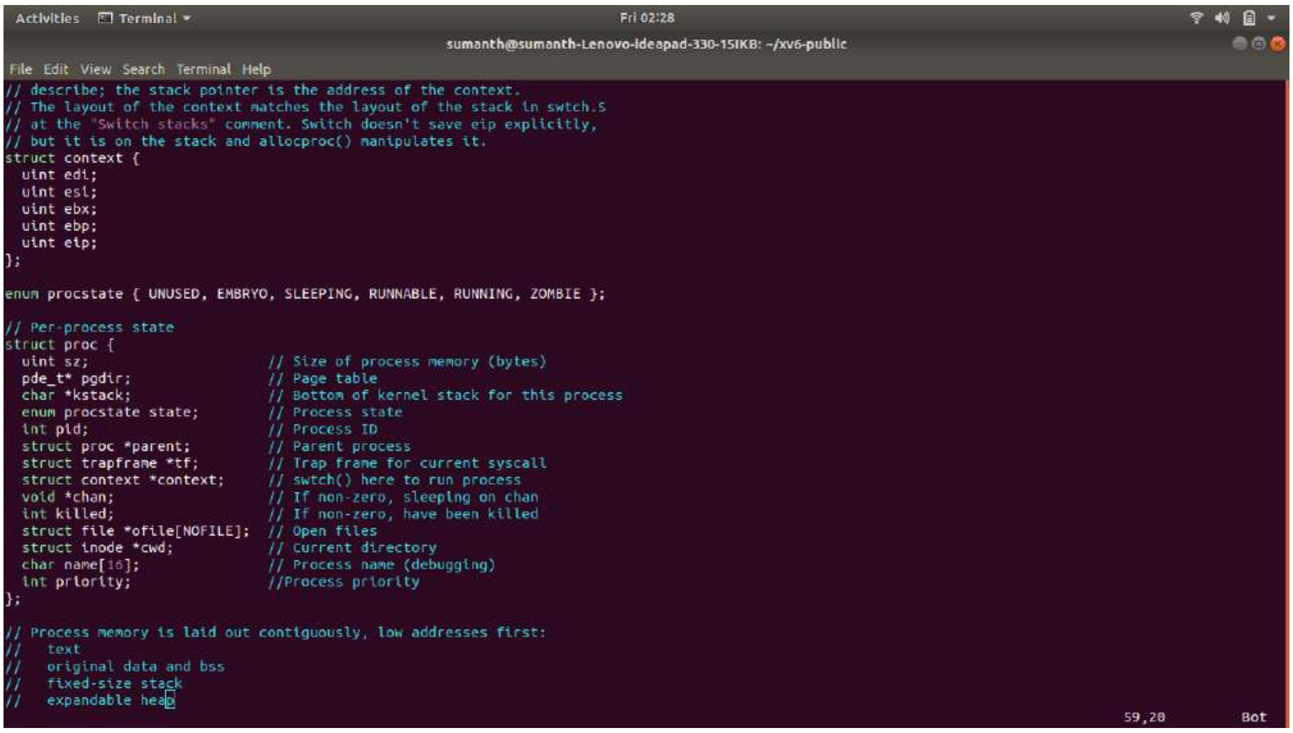
IMPLEMENTATION PRIORITY SCHEDULING

An overview of Priority Scheduling
Priority scheduling is one of the most common scheduling algorithms in batch systems. Priority scheduling is a method of scheduling processes based on priority. In this method, the scheduler chooses the tasks to work as per the priority. Each process is assigned a priority. Process with the highest priority is to be executed first and so on.

Processes with the same priority are executed on first come first served basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

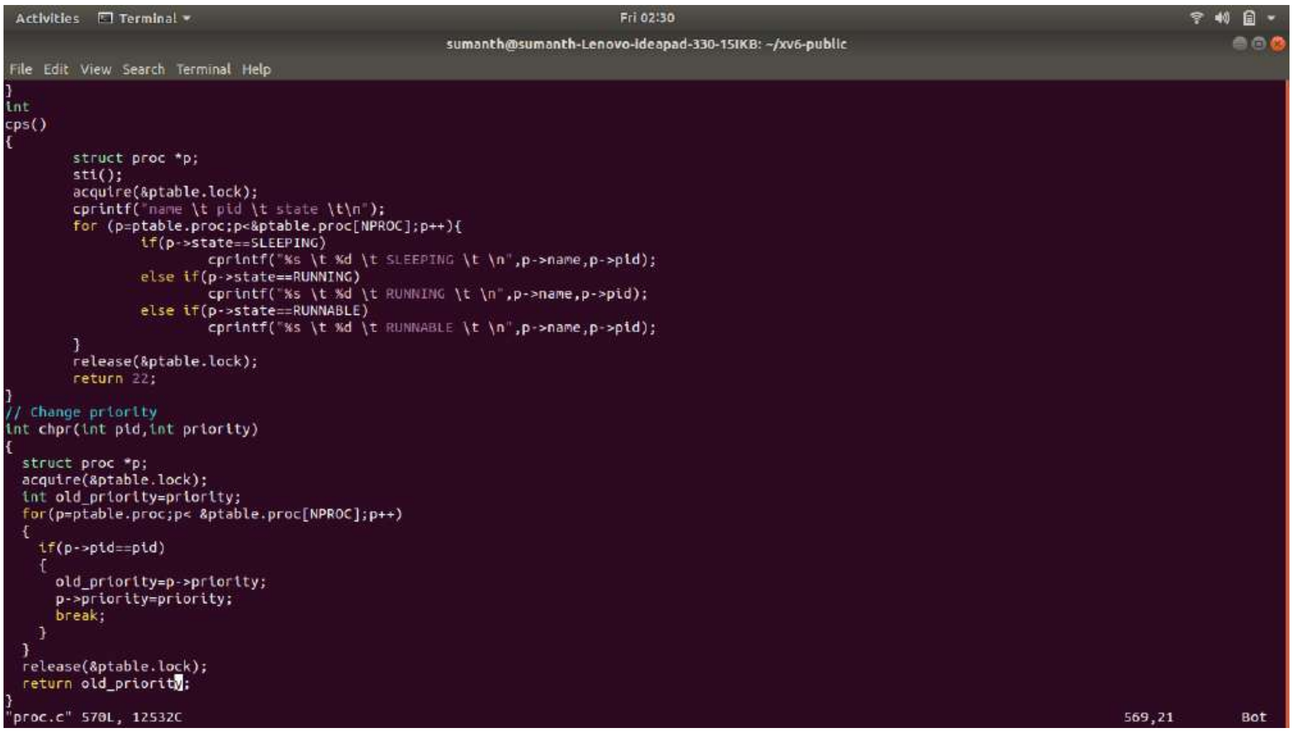
CONFIGURING PRIORITY FOR EVERY RUNNING PROCESS

1. Add priority to struct proc.h


Activities
Terminal
Fri 02:28
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

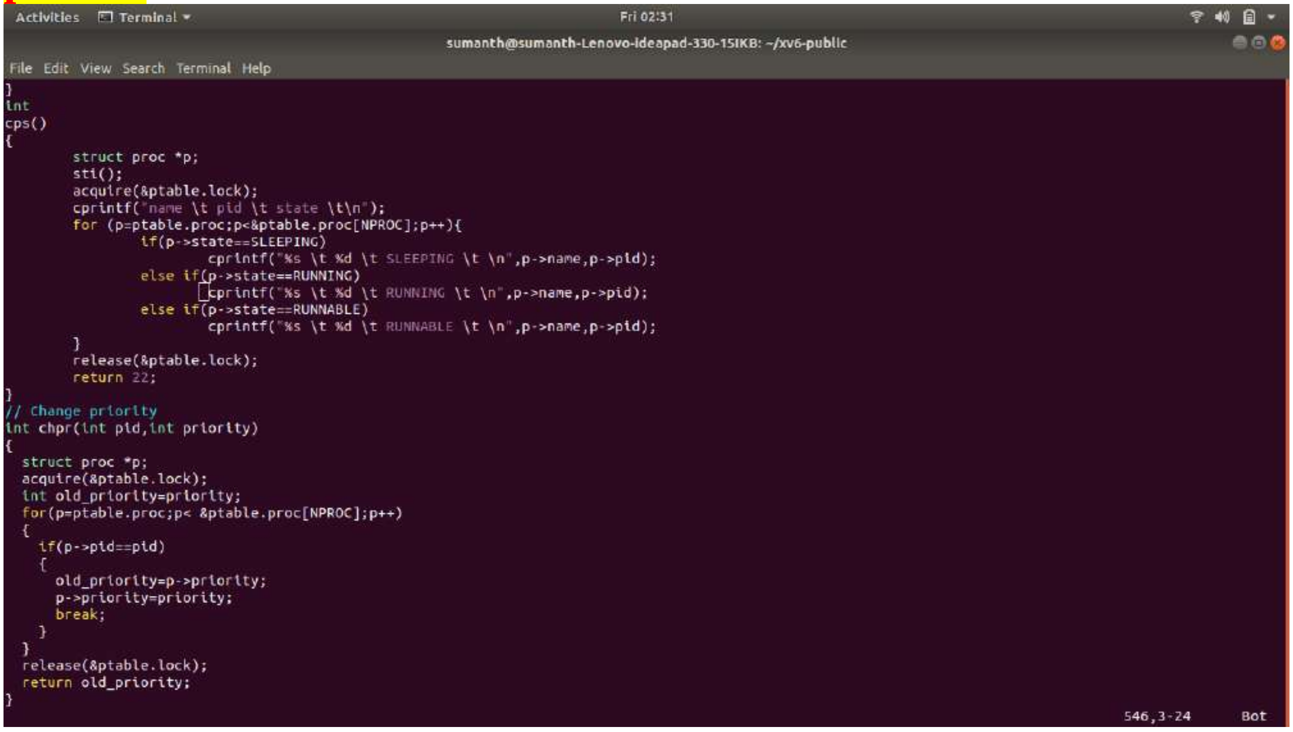
```
// describe; the stack pointer is the address of the context.
// The layout of the context matches the layout of the stack in swtch.S
// at the "Switch stacks" comment. Switch doesn't save eip explicitly,
// but it is on the stack and allocproc() manipulates it.
struct context {
  uint edi;
  uint esi;
  uint ebx;
  uint ebp;
  uint eip;
};

enum procstate { UNUSED, EMBRYO, SLEEPING, RUNNABLE, RUNNING, ZOMBIE };

// Per-process state
struct proc {
  uint sz;                     // Size of process memory (bytes)
  pde_t* pgdir;                // Page table
  char *kstack;                // Bottom of kernel stack for this process
  enum procstate state;        // Process state
  int pid;                     // Process ID
  struct proc *parent;         // Parent process
  struct trapframe *tf;        // Trap frame for current syscall
  struct context *context;     // swtch() here to run process
  void *chan;                  // If non-zero, sleeping on chan
  int killed;                  // If non-zero, have been killed
  struct file *ofile[NOFILE];  // Open files
  struct inode *cwd;           // Current directory
  char name[16];               // Process name (debugging)
  int priority;                //Process priority
};

// Process memory is laid out contiguously, low addresses first:
//   text
//   original data and bss
//   fixed-size stack
//   expandable heap
                                                                  59,28          Bot
```

2. Assign default priority in allocproc() in proc.c:


Activities
Terminal
Fri 02:30
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

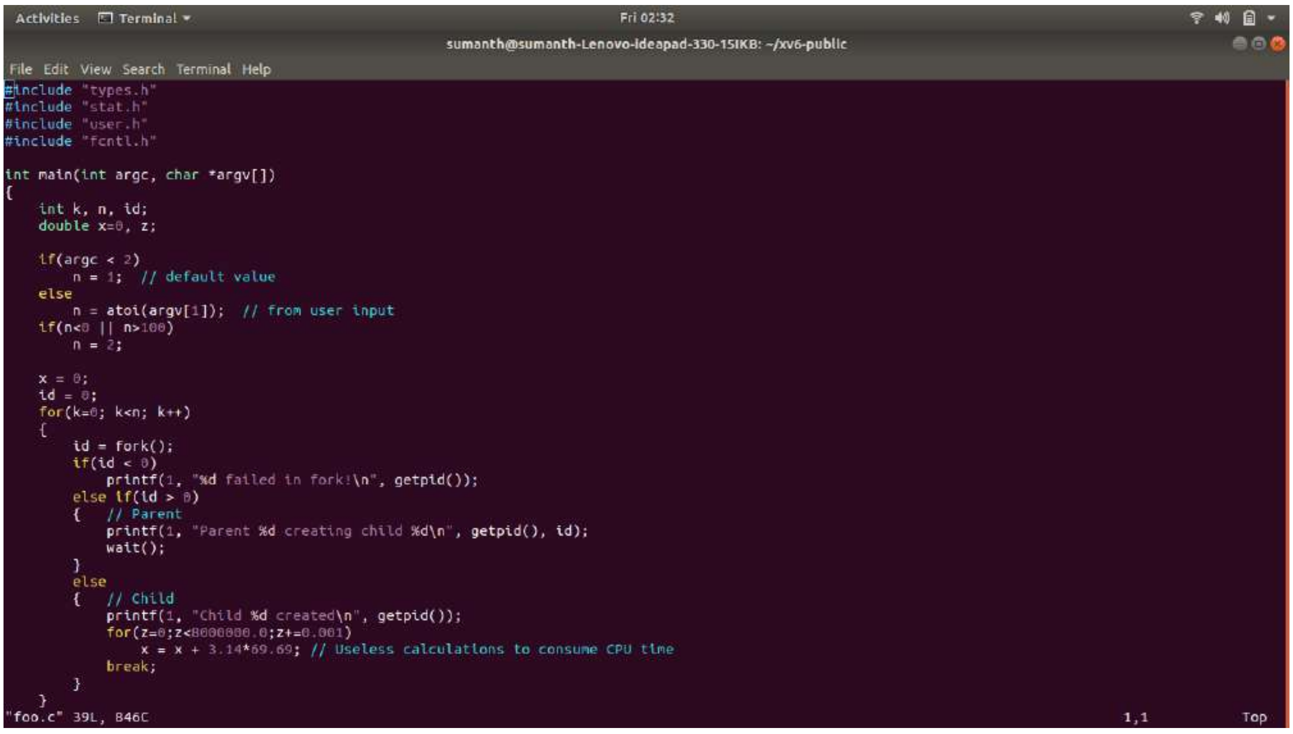
```
}
int
cps()
{
        struct proc *p;
        sti();
        acquire(&ptable.lock);
        cprintf("name \t pid \t state \t\n");
        for (p=ptable.proc;p<&ptable.proc[NPROC];p++){
                if(p->state==SLEEPING)
                        cprintf("%s \t %d \t SLEEPING \t \n",p->name,p->pid);
                else if(p->state==RUNNING)
                        cprintf("%s \t %d \t RUNNING \t \n",p->name,p->pid);
                else if(p->state==RUNNABLE)
                        cprintf("%s \t %d \t RUNNABLE \t \n",p->name,p->pid);
        }
        release(&ptable.lock);
        return 22;
}
// Change priority
int chpr(int pid,int priority)
{
  struct proc *p;
  acquire(&ptable.lock);
  int old_priority=priority;
  for(p=ptable.proc;p< &ptable.proc[NPROC];p++)
  {
    if(p->pid==pid)
    {
      old_priority=p->priority;
      p->priority=priority;
      break;
    }
  }
  release(&ptable.lock);
  return old_priority;
}
"proc.c" 570L, 12532C                                                    569,21         Bot
```

3. Configure process priority to printout the priority of every process


Activities
Terminal
Fri 02:31
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

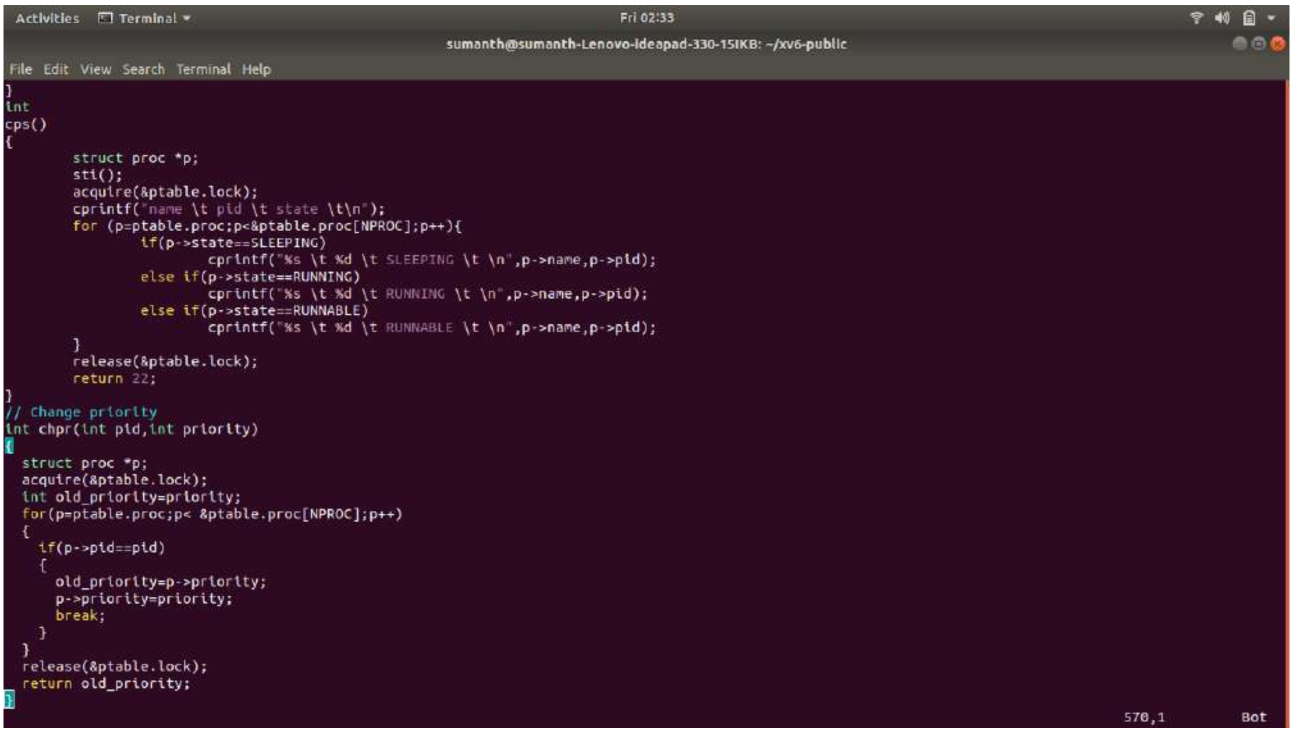
```
}
int
cps()
{
        struct proc *p;
        sti();
        acquire(&ptable.lock);
        cprintf("name \t pid \t state \t\n");
        for (p=ptable.proc;p<&ptable.proc[NPROC];p++){
                if(p->state==SLEEPING)
                        cprintf("%s \t %d \t SLEEPING \t \n",p->name,p->pid);
                else if(p->state==RUNNING)
                        cprintf("%s \t %d \t RUNNING \t \n",p->name,p->pid);
                else if(p->state==RUNNABLE)
                        cprintf("%s \t %d \t RUNNABLE \t \n",p->name,p->pid);
        }
        release(&ptable.lock);
        return 22;
}
// Change priority
int chpr(int pid,int priority)
{
  struct proc *p;
  acquire(&ptable.lock);
  int old_priority=priority;
  for(p=ptable.proc;p< &ptable.proc[NPROC];p++)
  {
    if(p->pid==pid)
    {
      old_priority=p->priority;
      p->priority=priority;
      break;
    }
  }
  release(&ptable.lock);
  return old_priority;
}
                                                                         546,3-24       Bot
```

4.Write a dummy program foo.c that creates child processes and consumes some computing time


Activities
Terminal
Fri 02:32
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

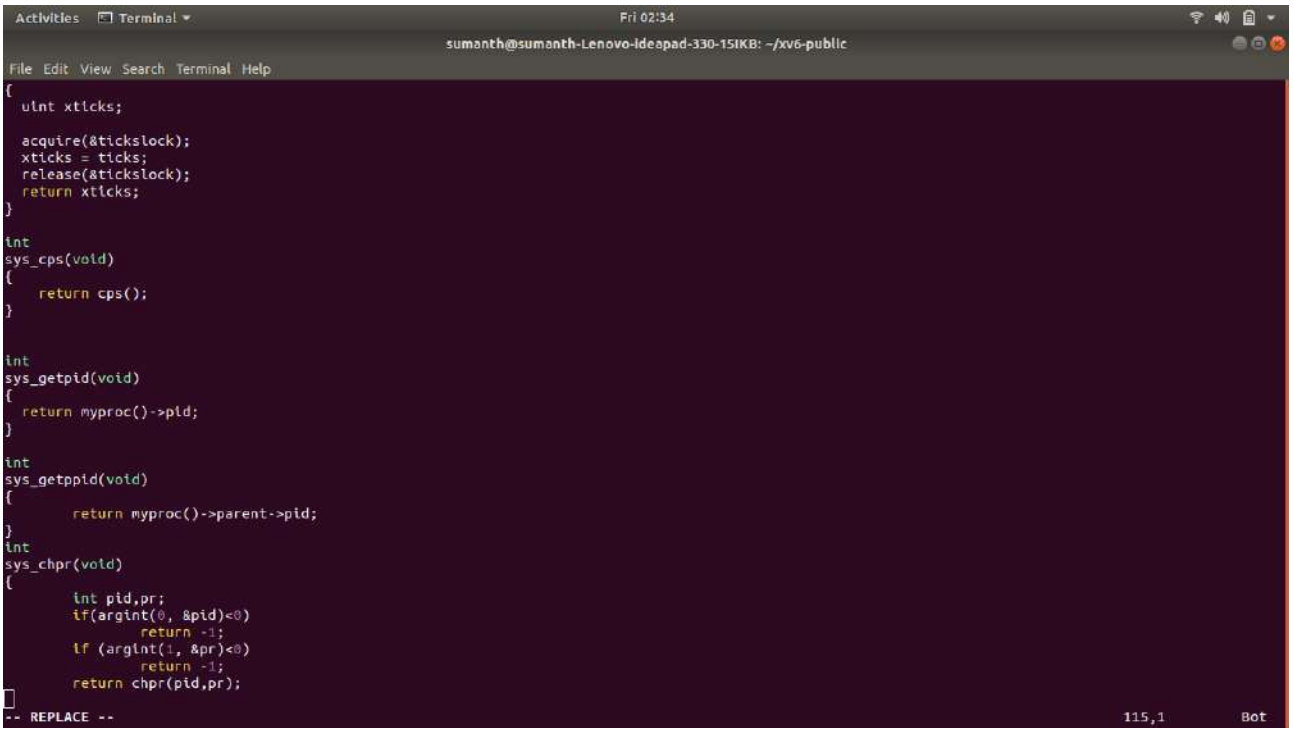
```
#include "types.h"
#include "stat.h"
#include "user.h"
#include "fcntl.h"

int main(int argc, char *argv[])
{
    int k, n, id;
    double x=0, z;

    if(argc < 2)
        n = 1;  // default value
    else
        n = atoi(argv[1]);  // from user input
    if(n<0 || n>100)
        n = 2;

    x = 0;
    id = 0;
    for(k=0; k<n; k++)
    {
        id = fork();
        if(id < 0)
            printf(1, "%d failed in fork!\n", getpid());
        else if(id > 0)
        {   // Parent
            printf(1, "Parent %d creating child %d\n", getpid(), id);
            wait();
        }
        else
        {   // Child
            printf(1, "Child %d created\n", getpid());
            for(z=0;z<8000000.0;z+=0.001)
                x = x + 3.14*69.69; // Useless calculations to consume CPU time
            break;
        }
    }
"foo.c" 39L, 846C                                                   1,1           Top
```

5. Add the function chpr() [change priority] to proc.c


Activities
Terminal
Fri 02:33
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

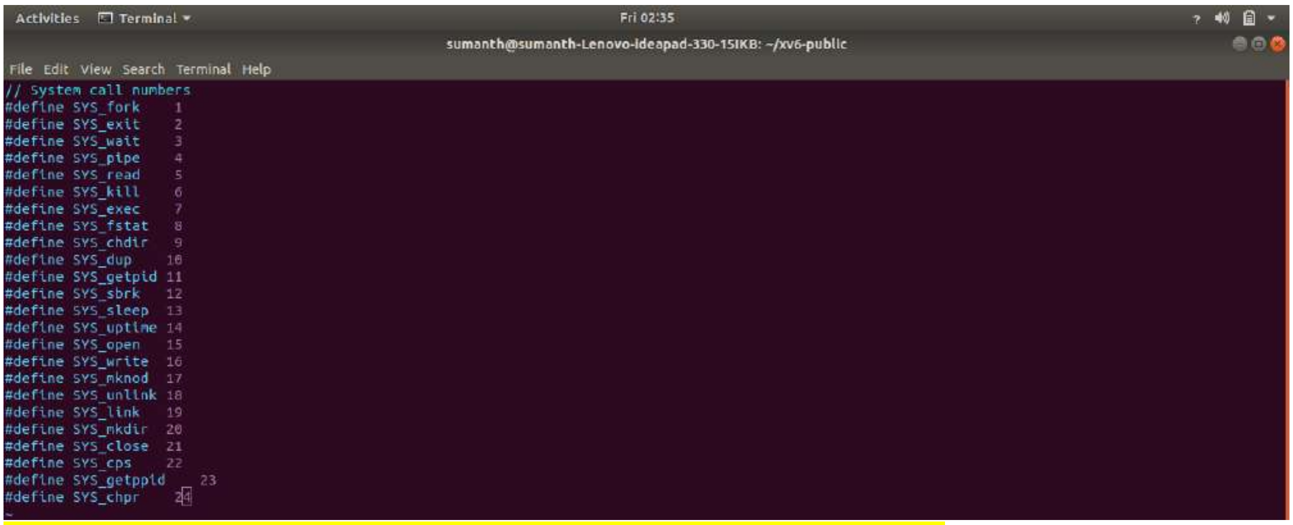
```
}
int
cps()
{
        struct proc *p;
        sti();
        acquire(&ptable.lock);
        cprintf("name \t pid \t state \t\n");
        for (p=ptable.proc;p<&ptable.proc[NPROC];p++){
                if(p->state==SLEEPING)
                        cprintf("%s \t %d \t SLEEPING \t \n",p->name,p->pid);
                else if(p->state==RUNNING)
                        cprintf("%s \t %d \t RUNNING \t \n",p->name,p->pid);
                else if(p->state==RUNNABLE)
                        cprintf("%s \t %d \t RUNNABLE \t \n",p->name,p->pid);
        }
        release(&ptable.lock);
        return 22;
}
// Change priority
int chpr(int pid,int priority)
{
  struct proc *p;
  acquire(&ptable.lock);
  int old_priority=priority;
  for(p=ptable.proc;p< &ptable.proc[NPROC];p++)
  {
    if(p->pid==pid)
    {
      old_priority=p->priority;
      p->priority=priority;
      break;
    }
  }
  release(&ptable.lock);
  return old_priority;
}
                                                                    570,1         Bot
```

6.Add sys_chpr to sysproc.c



Activities
Terminal
Fri 02:34
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
{
  uint xticks;

  acquire(&tickslock);
  xticks = ticks;
  release(&tickslock);
  return xticks;
}

int
sys_cps(void)
{
    return cps();
}


int
sys_getpid(void)
{
  return myproc()->pid;
}

int
sys_getppid(void)
{
        return myproc()->parent->pid;
}
int
sys_chpr(void)
{
        int pid,pr;
        if(argint(0, &pid)<0)
                return -1;
        if (argint(1, &pr)<0)
                return -1;
        return chpr(pid,pr);

-- REPLACE --                                                      115,1         Bot
```

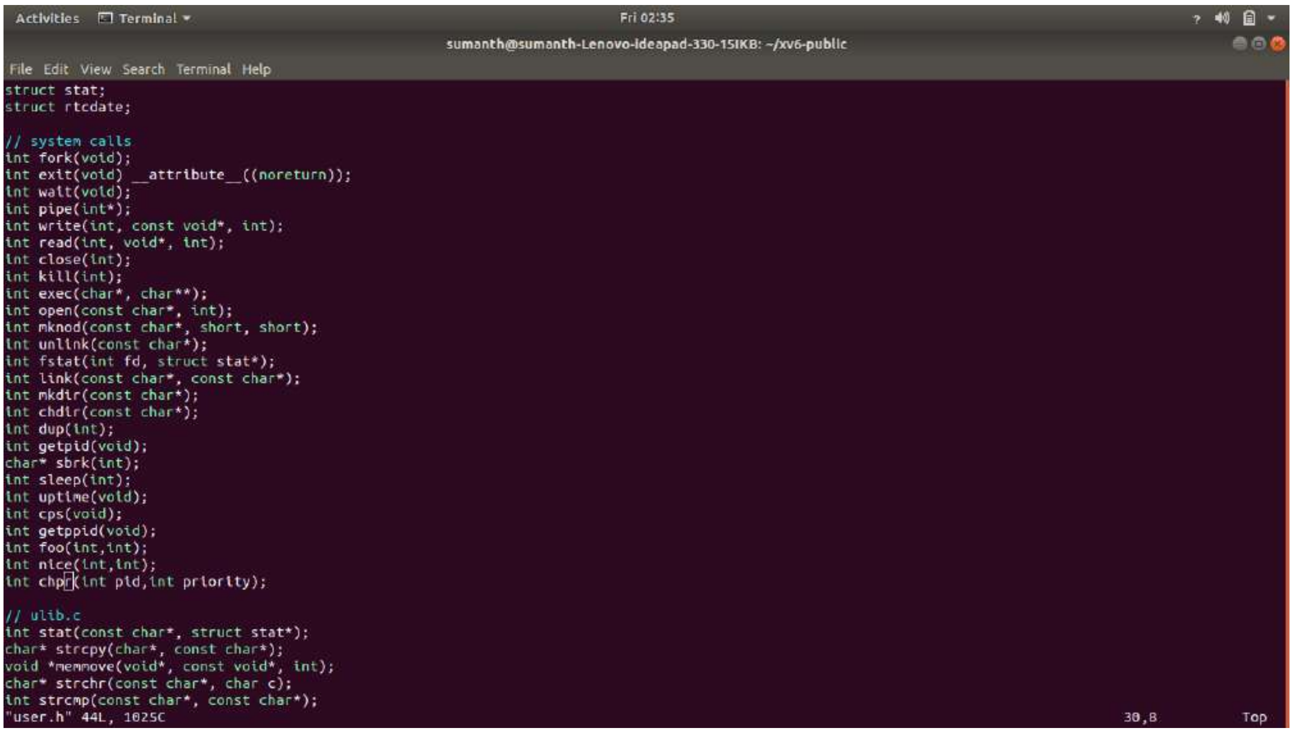


Activities
Terminal
Fri 02:35
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
// 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
#define SYS_unlink 18
#define SYS_link   19
#define SYS_mkdir  20
#define SYS_close  21
#define SYS_cps    22
#define SYS_getppid     23
#define SYS_chpr    24
```

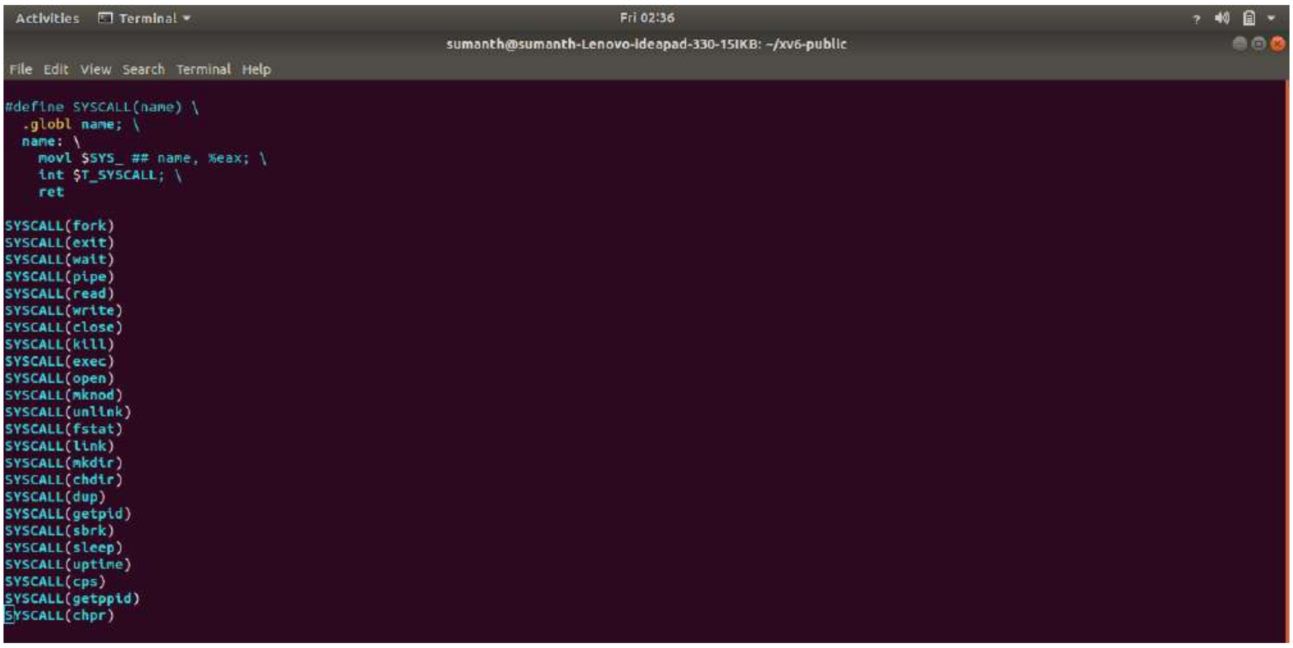
7.Adding chpr() as system call as done with cps()

Defining the syscall in user.h



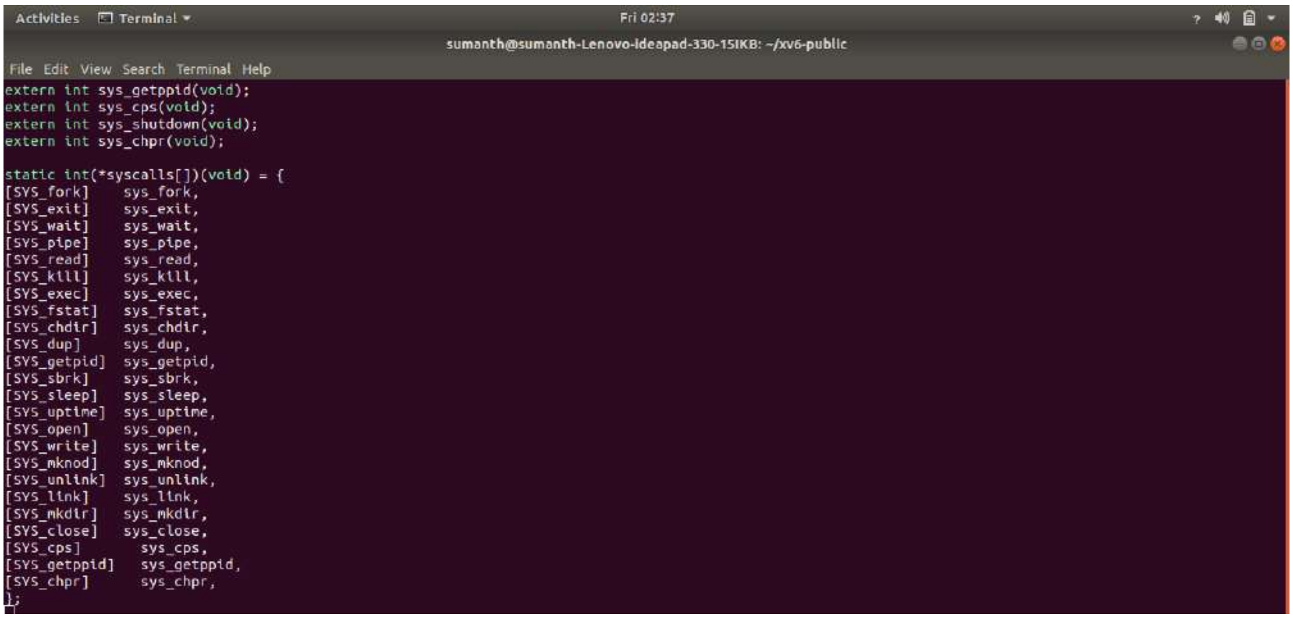
Activities
Terminal
Fri 02:35
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
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*);
int link(const char*, const char*);
int mkdir(const char*);
int chdir(const char*);
int dup(int);
int getpid(void);
char* sbrk(int);
int sleep(int);
int uptime(void);
int cps(void);
int getppid(void);
int foo(int,int);
int nice(int,int);
int chpr(int pid,int priority);

// ulib.c
int stat(const char*, struct stat*);
char* strcpy(char*, const char*);
void *memmove(void*, const void*, int);
char* strchr(const char*, char c);
int strcmp(const char*, const char*);
"user.h" 44L, 1025C                                                    30,8          Top
```

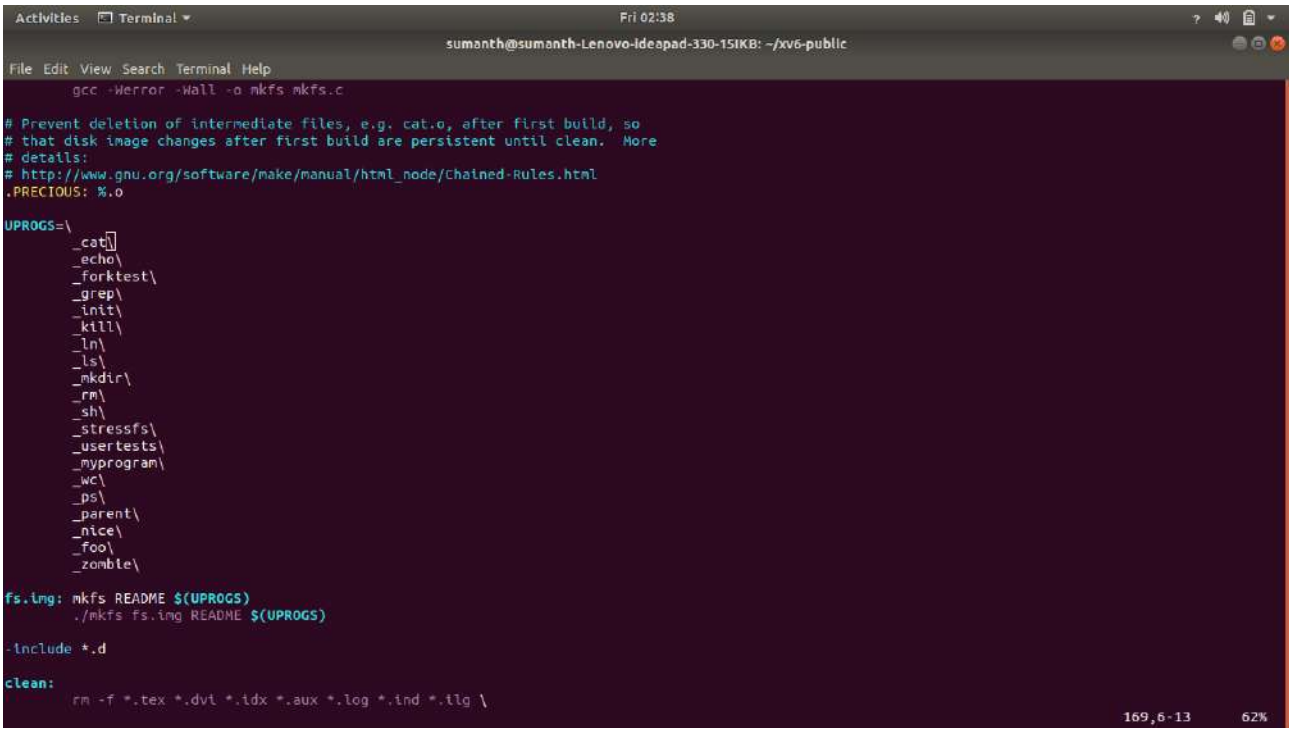


Activities
Terminal
Fri 02:36
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

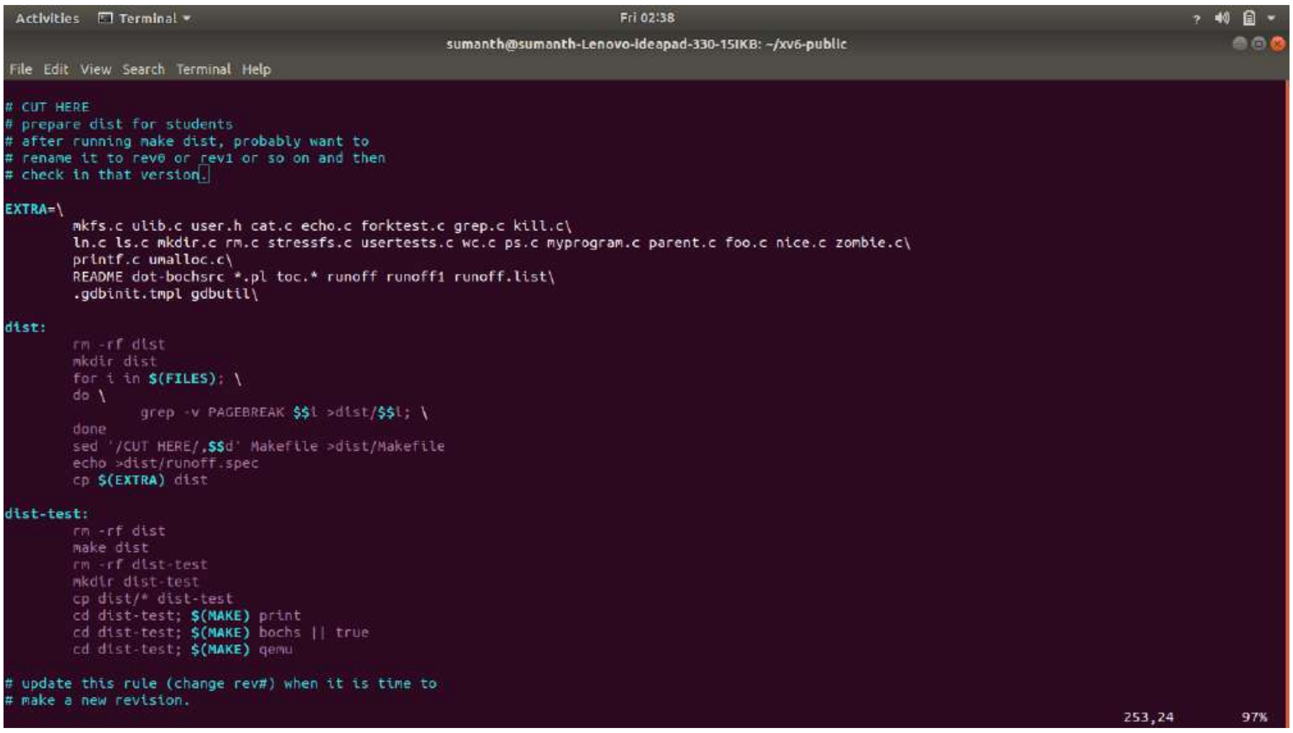
```
#define SYSCALL(name) \
  .globl name; \
  name: \
    movl $SYS_ ## name, %eax; \
    int $T_SYSCALL; \
    ret

SYSCALL(fork)
SYSCALL(exit)
SYSCALL(wait)
SYSCALL(pipe)
SYSCALL(read)
SYSCALL(write)
SYSCALL(close)
SYSCALL(kill)
SYSCALL(exec)
SYSCALL(open)
SYSCALL(mknod)
SYSCALL(unlink)
SYSCALL(fstat)
SYSCALL(link)
SYSCALL(mkdir)
SYSCALL(chdir)
SYSCALL(dup)
SYSCALL(getpid)
SYSCALL(sbrk)
SYSCALL(sleep)
SYSCALL(uptime)
SYSCALL(cps)
SYSCALL(getppid)
SYSCALL(chpr)
```


Activities
Terminal
Fri 02:37
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

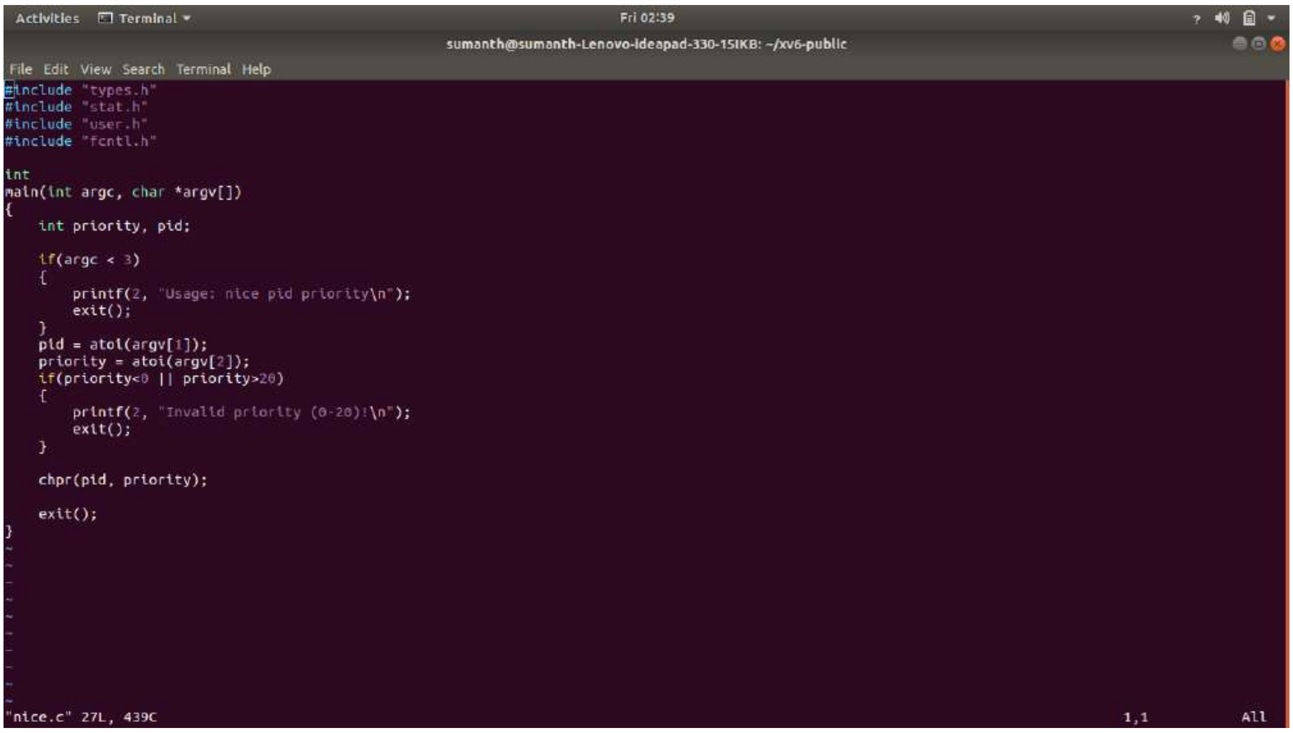
```
extern int sys_getppid(void);
extern int sys_cps(void);
extern int sys_shutdown(void);
extern int sys_chpr(void);

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,
[SYS_mknod]   sys_mknod,
[SYS_unlink]  sys_unlink,
[SYS_link]    sys_link,
[SYS_mkdir]   sys_mkdir,
[SYS_close]   sys_close,
[SYS_cps]       sys_cps,
[SYS_getppid]   sys_getppid,
[SYS_chpr]      sys_chpr,
};
```


Activities
Terminal
Fri 02:38
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
        gcc -Werror -Wall -o mkfs mkfs.c

# Prevent deletion of intermediate files, e.g. cat.o, after first build, so
# that disk image changes after first build are persistent until clean.  More
# details:
# http://www.gnu.org/software/make/manual/html_node/Chained-Rules.html
.PRECIOUS: %.o

UPROGS=\
        _cat\
        _echo\
        _forktest\
        _grep\
        _init\
        _kill\
        _ln\
        _ls\
        _mkdir\
        _rm\
        _sh\
        _stressfs\
        _usertests\
        _myprogram\
        _wc\
        _ps\
        _parent\
        _nice\
        _foo\
        _zombie\

fs.img: mkfs README $(UPROGS)
        ./mkfs fs.img README $(UPROGS)

-include *.d

clean:
        rm -f *.tex *.dvi *.idx *.aux *.log *.ind *.ilg \
                                                                                    169,6-13      62%
```

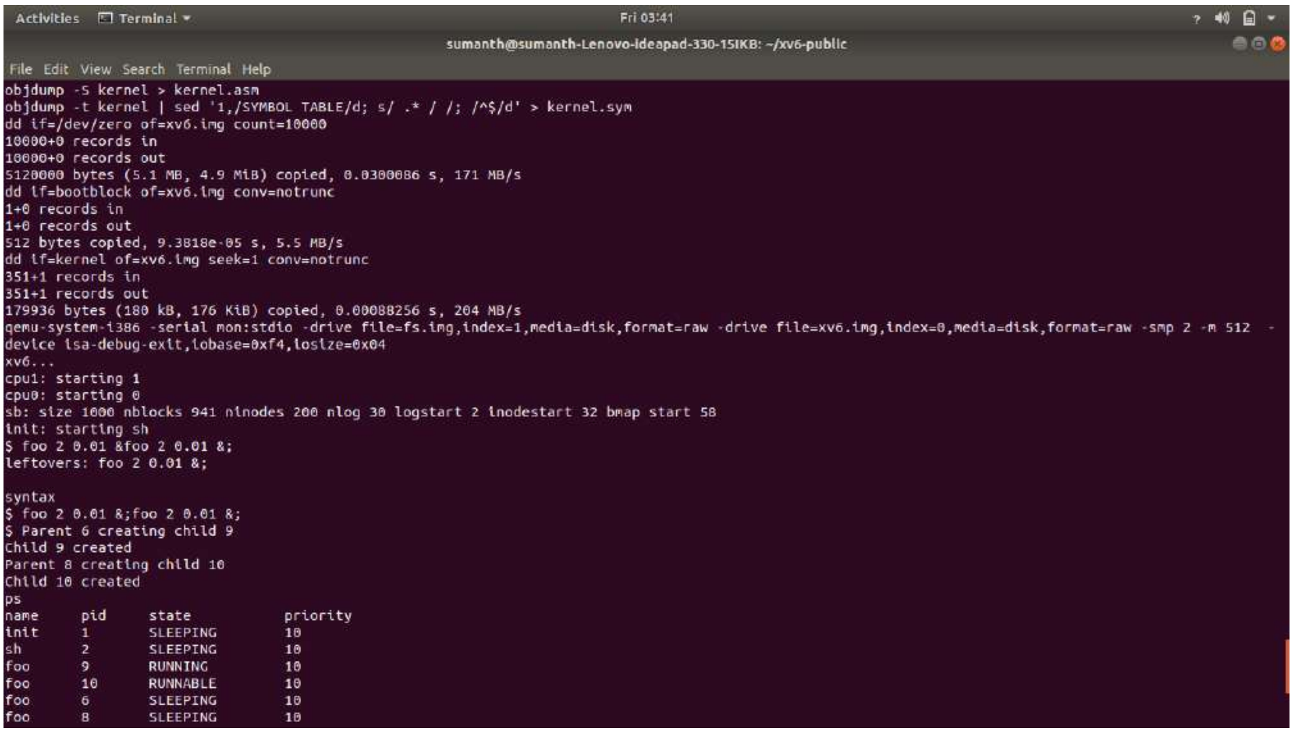


```
# CUT HERE
# prepare dist for students
# after running make dist, probably want to
# rename it to rev0 or rev1 or so on and then
# check in that version.

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 ps.c myprogram.c parent.c foo.c nice.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 \
		grep -v PAGEBREAK $$i >dist/$$i; \
	done
	sed '/CUT HERE/,$$d' Makefile >dist/Makefile
	echo >dist/runoff.spec
	cp $(EXTRA) dist

dist-test:
	rm -rf dist
	make dist
	rm -rf dist-test
	mkdir dist-test
	cp dist/* dist-test
	cd dist-test; $(MAKE) print
	cd dist-test; $(MAKE) bochs || true
	cd dist-test; $(MAKE) qemu

# update this rule (change rev#) when it is time to
# make a new revision.
                                                                    253,24        97%
```

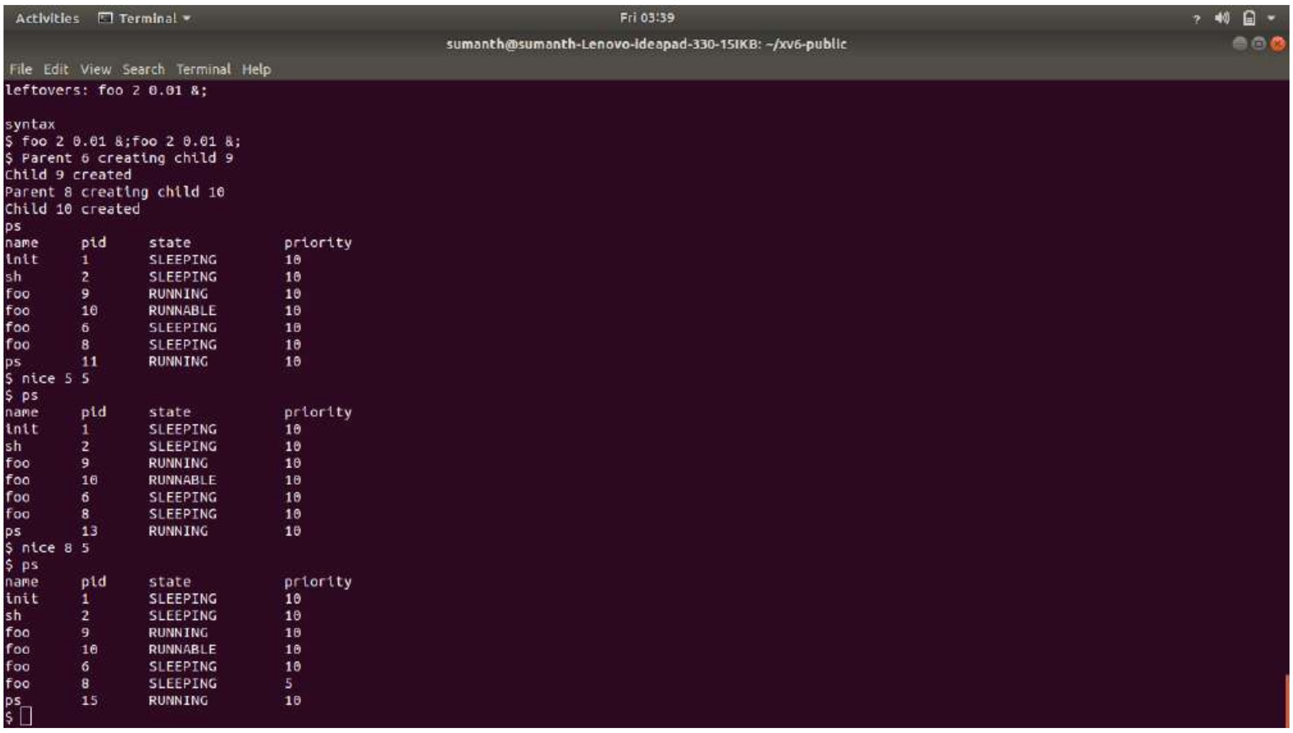


```
#include "types.h"
#include "stat.h"
#include "user.h"
#include "fcntl.h"

int
main(int argc, char *argv[])
{
    int priority, pid;

    if(argc < 3)
    {
        printf(2, "Usage: nice pid priority\n");
        exit();
    }
    pid = atoi(argv[1]);
    priority = atoi(argv[2]);
    if(priority<0 || priority>20)
    {
        printf(2, "Invalid priority (0-20)!\n");
        exit();
    }

    chpr(pid, priority);

    exit();
}
~
~
~
~
~
~
~
~
~
~
"nice.c" 27L, 439C                                                  1,1           All
```

Creating two child processes using foo system call:


Activities
Terminal
Fri 03:41
sumanth@sumanth-Lenovo-Ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

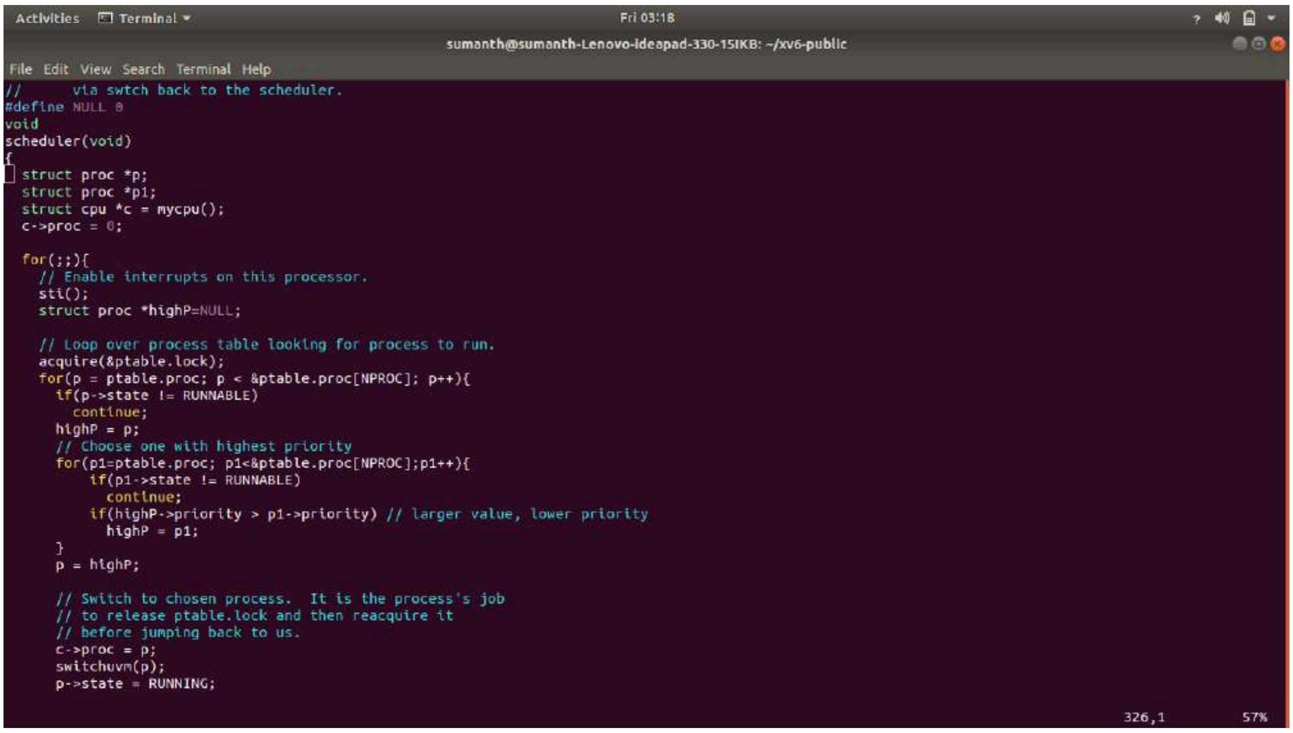
```
objdump -S kernel > kernel.asm
objdump -t kernel | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > kernel.sym
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.0300086 s, 171 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 9.3818e-05 s, 5.5 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
351+1 records in
351+1 records out
179936 bytes (180 kB, 176 KiB) copied, 0.00088256 s, 204 MB/s
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512  -
device isa-debug-exit,iobase=0xf4,iosize=0x04
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ foo 2 0.01 &foo 2 0.01 &;
leftovers: foo 2 0.01 &;

syntax
$ foo 2 0.01 &;foo 2 0.01 &;
$ Parent 6 creating child 9
Child 9 created
Parent 8 creating child 10
Child 10 created
ps
name    pid     state           priority
init    1       SLEEPING        10
sh      2       SLEEPING        10
foo     9       RUNNING         10
foo     10      RUNNABLE        10
foo     6       SLEEPING        10
foo     8       SLEEPING        10
```


Activities
Terminal
Fri 03:39
sumanth@sumanth-Lenovo-Ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
leftovers: foo 2 0.01 &;

syntax
$ foo 2 0.01 &;foo 2 0.01 &;
$ Parent 6 creating child 9
Child 9 created
Parent 8 creating child 10
Child 10 created
ps
name    pid     state           priority
init    1       SLEEPING        10
sh      2       SLEEPING        10
foo     9       RUNNING         10
foo     10      RUNNABLE        10
foo     6       SLEEPING        10
foo     8       SLEEPING        10
ps      11      RUNNING         10
$ nice 5 5
$ ps
name    pid     state           priority
init    1       SLEEPING        10
sh      2       SLEEPING        10
foo     9       RUNNING         10
foo     10      RUNNABLE        10
foo     6       SLEEPING        10
foo     8       SLEEPING        10
ps      13      RUNNING         10
$ nice 8 5
$ ps
name    pid     state           priority
init    1       SLEEPING        10
sh      2       SLEEPING        10
foo     9       RUNNING         10
foo     10      RUNNABLE        10
foo     6       SLEEPING        10
foo     8       SLEEPING        5
ps      15      RUNNING         10
$ 
```

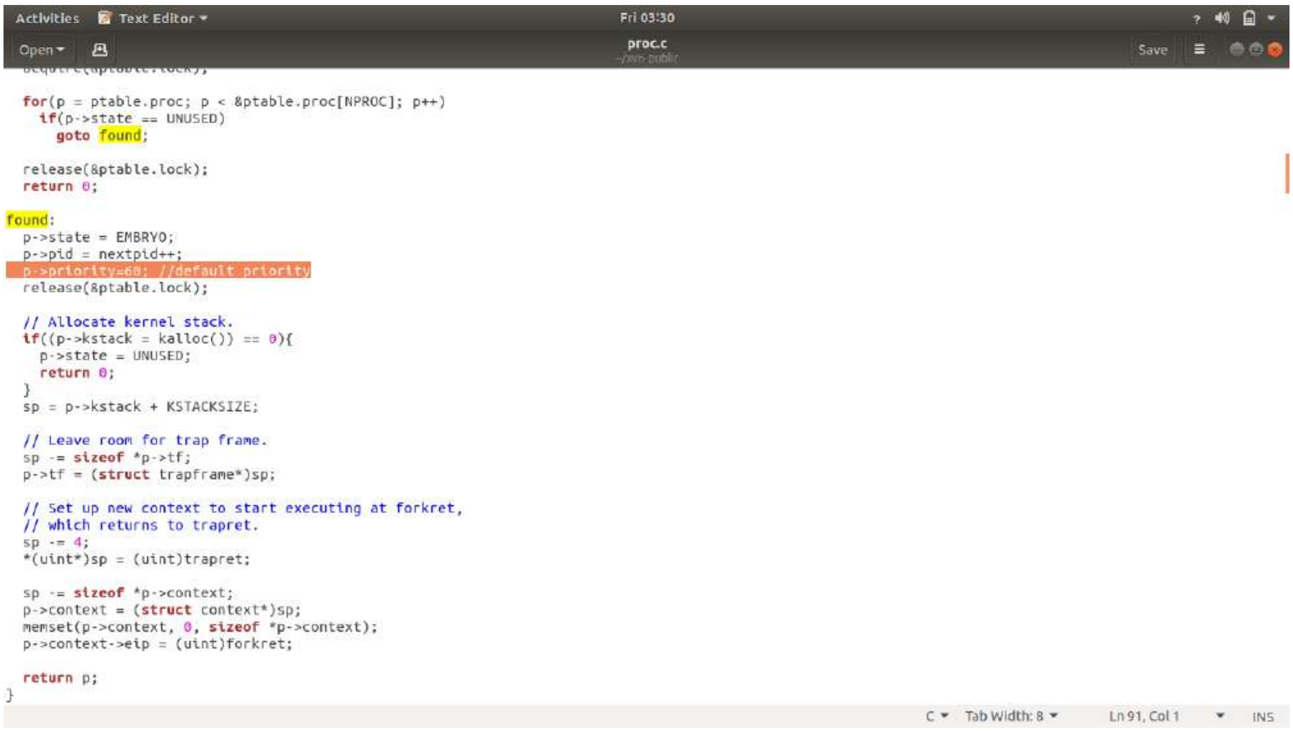
IMPLEMENT PRIORITY SCHEDULING

Modifying proc.c scheduler function to change the default scheduling from RR to Priority scheduling


Activities
Terminal
Fri 03:18
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help
326,1
57%

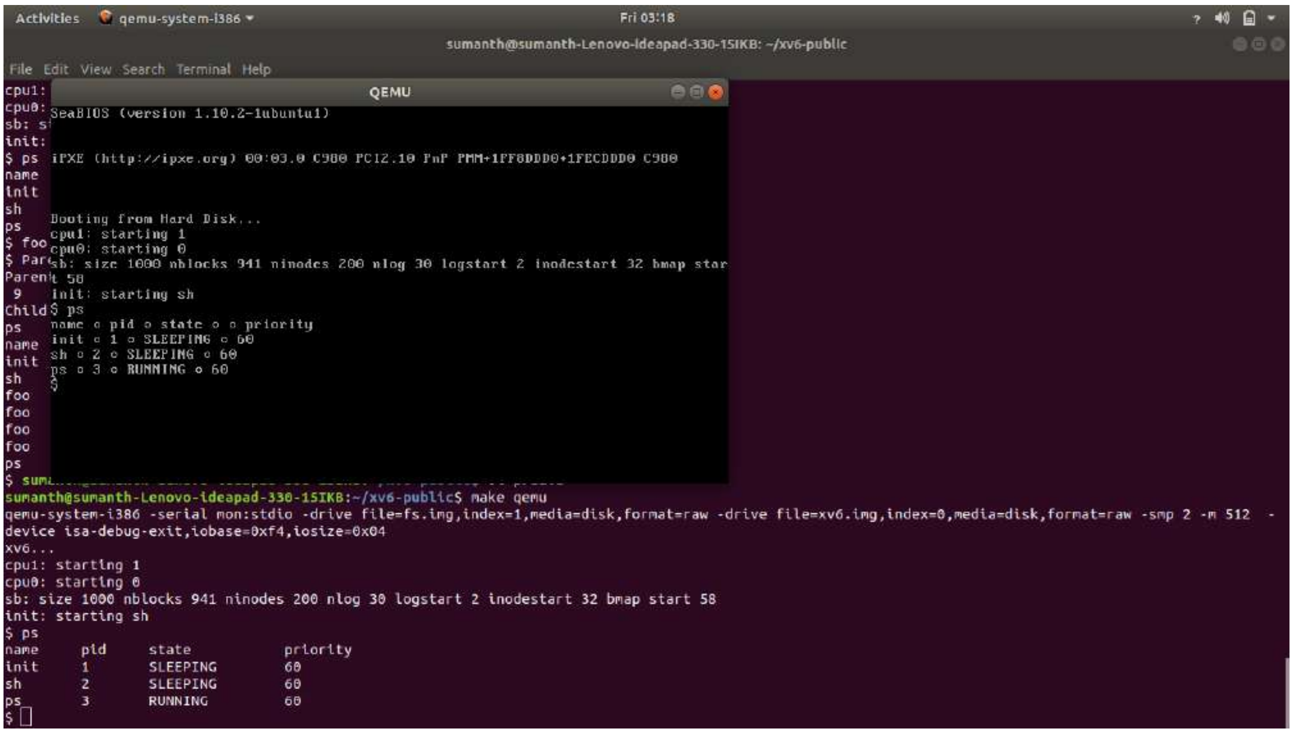
```
//      via swtch back to the scheduler.
#define NULL 0
void
scheduler(void)
{
  struct proc *p;
  struct proc *p1;
  struct cpu *c = mycpu();
  c->proc = 0;

  for(;;){
    // Enable interrupts on this processor.
    sti();
    struct proc *highP=NULL;

    // Loop over process table looking for process to run.
    acquire(&ptable.lock);
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
      if(p->state != RUNNABLE)
        continue;
      highP = p;
      // Choose one with highest priority
      for(p1=ptable.proc; p1<&ptable.proc[NPROC];p1++){
          if(p1->state != RUNNABLE)
            continue;
          if(highP->priority > p1->priority) // larger value, lower priority
            highP = p1;
      }
      p = highP;

      // Switch to chosen process.  It is the process's job
      // to release ptable.lock and then reacquire it
      // before jumping back to us.
      c->proc = p;
      switchuvm(p);
      p->state = RUNNING;
```

Modifying Default Priority In proc.c



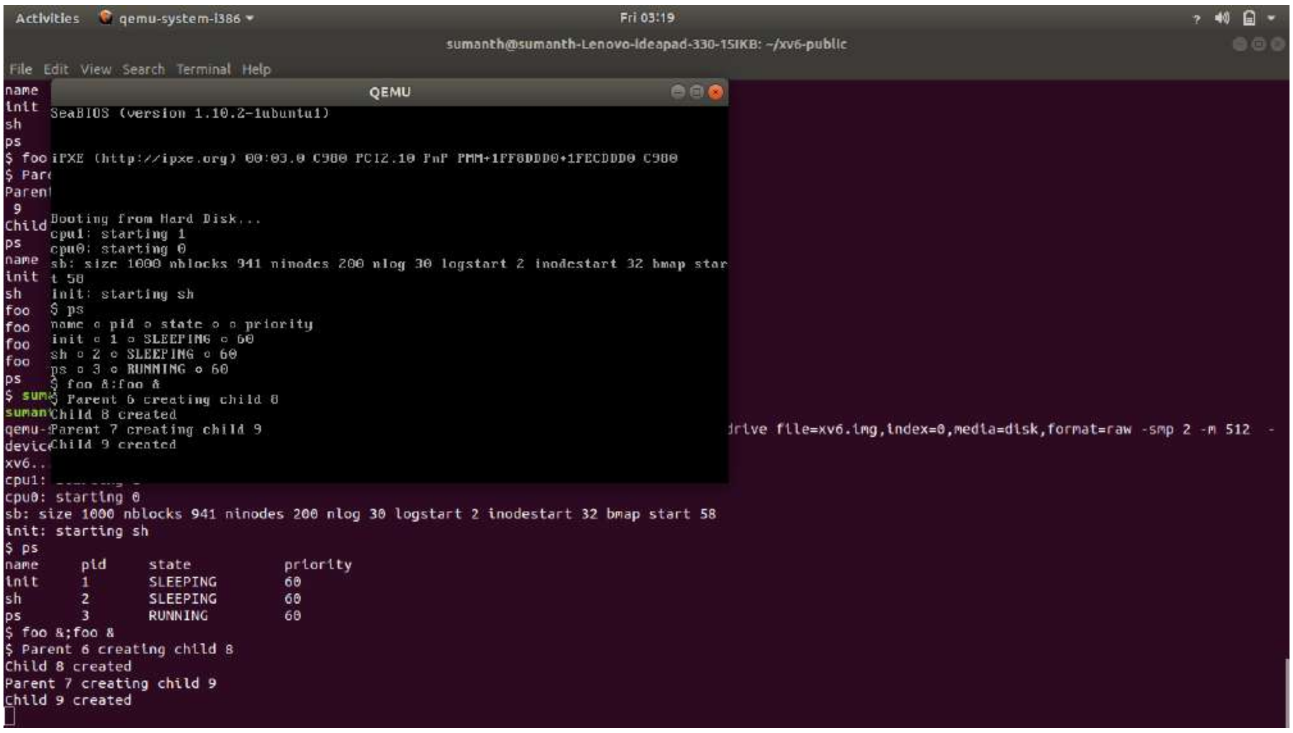
```c
  for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)
    if(p->state == UNUSED)
      goto found;

  release(&ptable.lock);
  return 0;

found:
  p->state = EMBRYO;
  p->pid = nextpid++;
  p->priority=60; //default priority
  release(&ptable.lock);

  // Allocate kernel stack.
  if((p->kstack = kalloc()) == 0){
    p->state = UNUSED;
    return 0;
  }
  sp = p->kstack + KSTACKSIZE;

  // Leave room for trap frame.
  sp -= sizeof *p->tf;
  p->tf = (struct trapframe*)sp;

  // Set up new context to start executing at forkret,
  // which returns to trapret.
  sp -= 4;
  *(uint*)sp = (uint)trapret;

  sp -= sizeof *p->context;
  p->context = (struct context*)sp;
  memset(p->context, 0, sizeof *p->context);
  p->context->eip = (uint)forkret;

  return p;
}
```

Create child processes by running foo system call



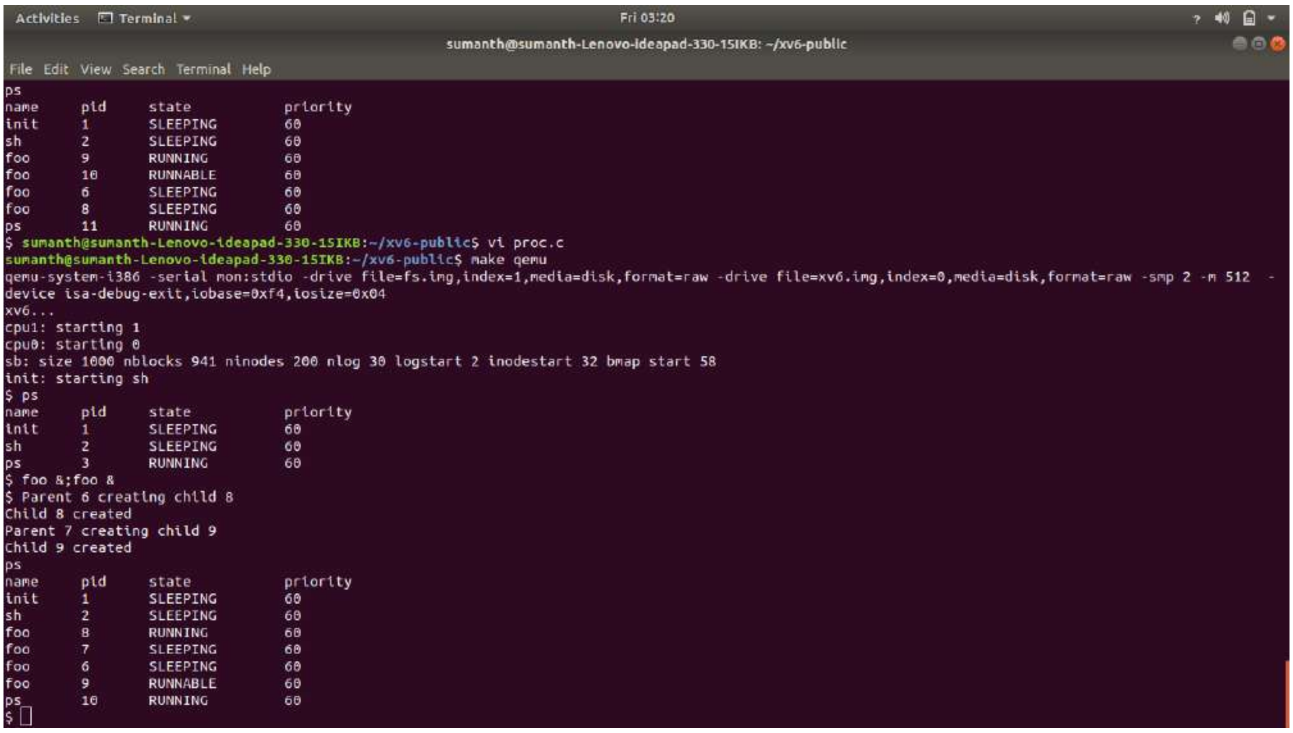
```
SeaBIOS (version 1.10.2-1ubuntu1)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF8DDD0+1FECDDD0 C980

Booting from Hard Disk...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ps
name o pid o state o o priority
init o 1 o SLEEPING o 60
sh o 2 o SLEEPING o 60
ps o 3 o RUNNING o 60
$
```

```
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ make qemu
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512 -device isa-debug-exit,iobase=0xf4,iosize=0x04
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ps
name    pid     state           priority
init    1       SLEEPING        60
sh      2       SLEEPING        60
ps      3       RUNNING         60
$ 
```



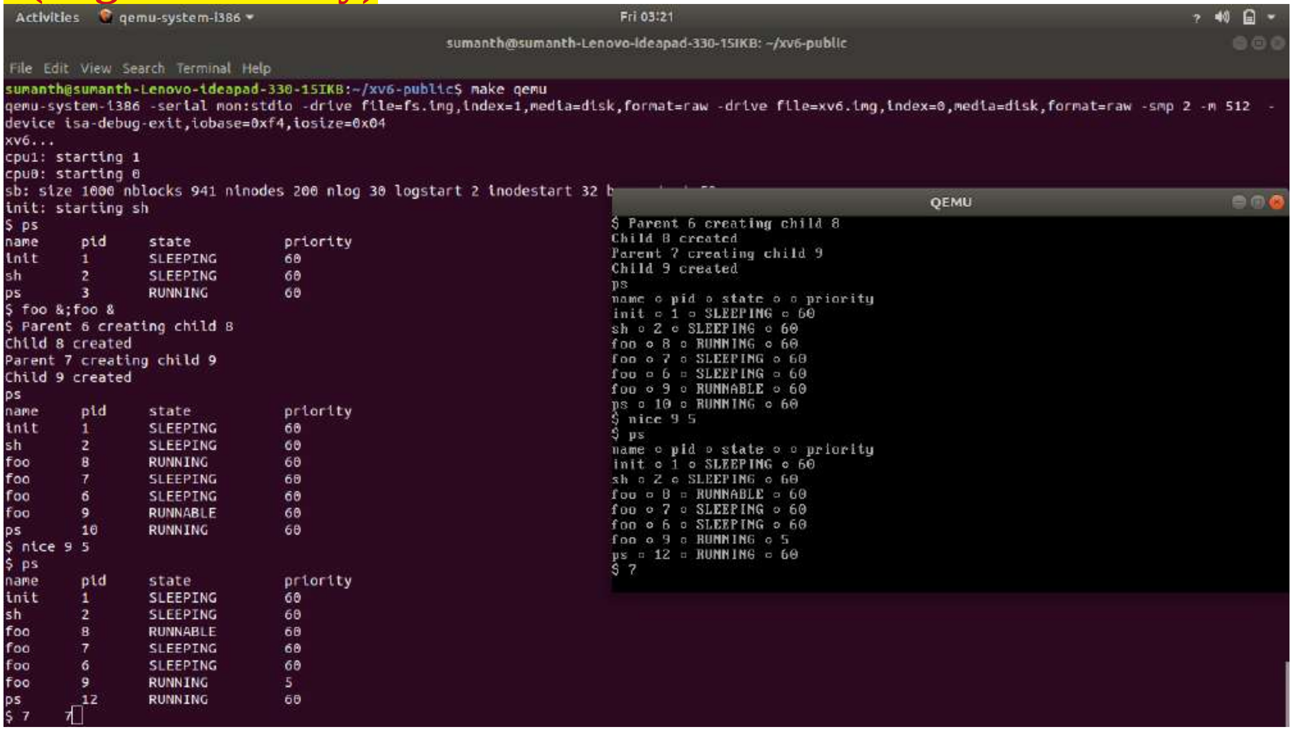
```
SeaBIOS (version 1.10.2-1ubuntu1)


iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF8DDD0+1FECDDD0 C980



Booting from Hard Disk...
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
$ ps
name ◦ pid ◦ state ◦ ◦ priority
init ◦ 1 ◦ SLEEPING ◦ 60
sh ◦ 2 ◦ SLEEPING ◦ 60
ps ◦ 3 ◦ RUNNING ◦ 60
$ foo &;foo &
$ Parent 6 creating child 8
Child 8 created
Parent 7 creating child 9
Child 9 created
```

```
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ps
name     pid     state           priority
init     1       SLEEPING        60
sh       2       SLEEPING        60
ps       3       RUNNING         60
$ foo &;foo &
$ Parent 6 creating child 8
Child 8 created
Parent 7 creating child 9
Child 9 created
```



```
ps
name     pid     state           priority
init     1       SLEEPING        60
sh       2       SLEEPING        60
foo      9       RUNNING         60
foo      10      RUNNABLE        60
foo      6       SLEEPING        60
foo      8       SLEEPING        60
ps       11      RUNNING         60
$ sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ vi proc.c
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ make qemu
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512  -
device isa-debug-exit,iobase=0xf4,iosize=0x04
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ps
name     pid     state           priority
init     1       SLEEPING        60
sh       2       SLEEPING        60
ps       3       RUNNING         60
$ foo &;foo &
$ Parent 6 creating child 8
Child 8 created
Parent 7 creating child 9
Child 9 created
ps
name     pid     state           priority
init     1       SLEEPING        60
sh       2       SLEEPING        60
foo      8       RUNNING         60
foo      7       SLEEPING        60
foo      6       SLEEPING        60
foo      9       RUNNABLE        60
ps       10      RUNNING         60
$
```

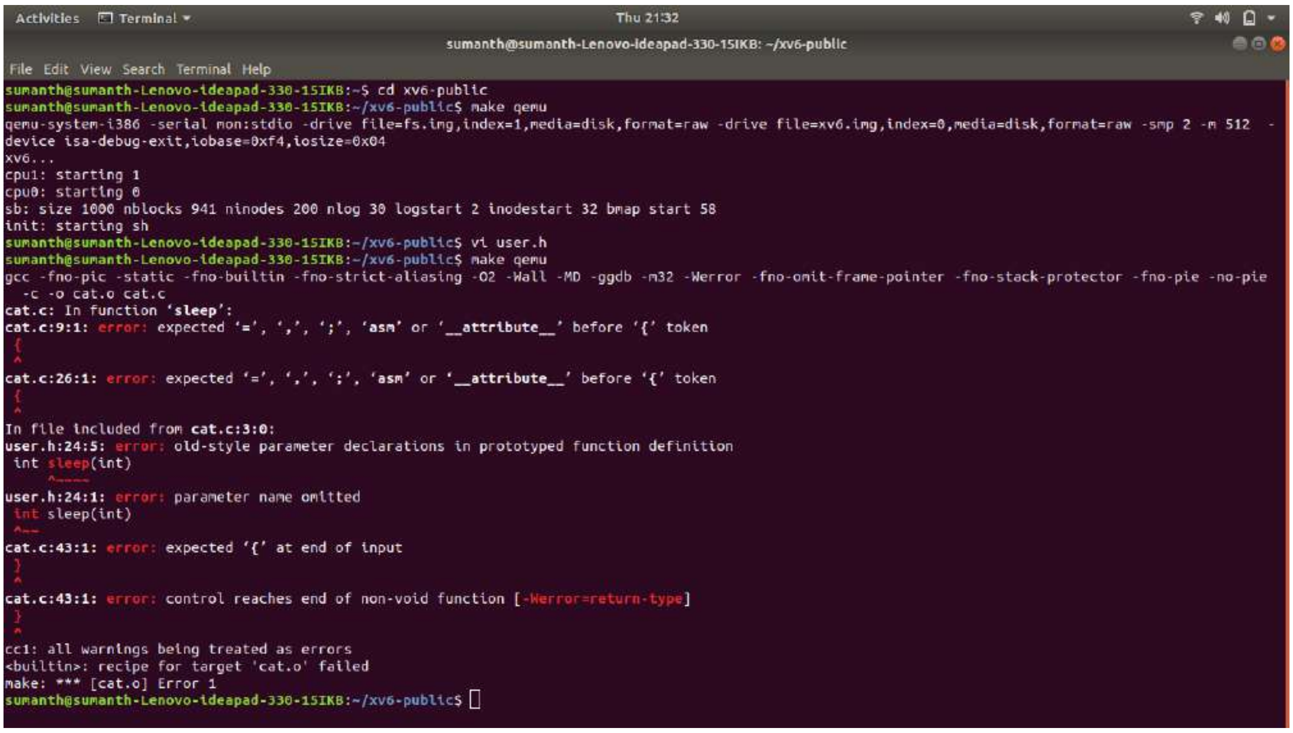
Change the priority of the process with pid 11 from 10 to 8(Higher Priority)


Activities
qemu-system-i386
Fri 03:21
sumanth@sumanth-Lenovo-Ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help
QEMU

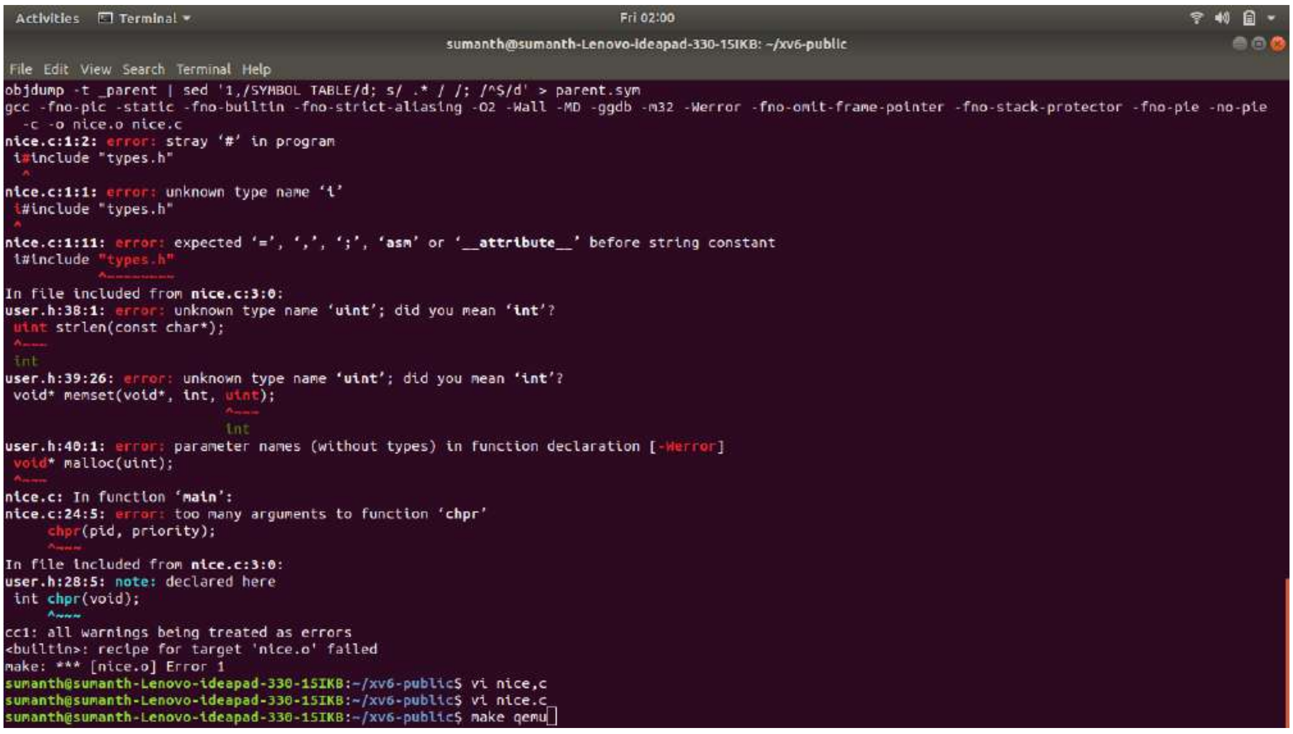
```
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ make qemu
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512  -
device isa-debug-exit,iobase=0xf4,iosize=0x04
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 b
init: starting sh
$ ps
name     pid      state          priority
init     1        SLEEPING       60
sh       2        SLEEPING       60
ps       3        RUNNING        60
$ foo &;foo &
$ Parent 6 creating child 8
Child 8 created
Parent 7 creating child 9
Child 9 created
ps
name     pid      state          priority
init     1        SLEEPING       60
sh       2        SLEEPING       60
foo      8        RUNNING        60
foo      7        SLEEPING       60
foo      6        SLEEPING       60
foo      9        RUNNABLE       60
ps       10       RUNNING        60
$ nice 9 5
$ ps
name     pid      state          priority
init     1        SLEEPING       60
sh       2        SLEEPING       60
foo      8        RUNNABLE       60
foo      7        SLEEPING       60
foo      6        SLEEPING       60
foo      9        RUNNING        5
ps       12       RUNNING        60
$ 7     7
```

```
$ Parent 6 creating child 8
Child 8 created
Parent 7 creating child 9
Child 9 created
ps
name ◦ pid ◦ state ◦ ◦ priority
init ◦ 1 ◦ SLEEPING ◦ 60
sh ◦ 2 ◦ SLEEPING ◦ 60
foo ◦ 8 ◦ RUNNING ◦ 60
foo ◦ 7 ◦ SLEEPING ◦ 60
foo ◦ 6 ◦ SLEEPING ◦ 60
foo ◦ 9 ◦ RUNNABLE ◦ 60
ps ◦ 10 ◦ RUNNING ◦ 60
$ nice 9 5
$ ps
name ◦ pid ◦ state ◦ ◦ priority
init ◦ 1 ◦ SLEEPING ◦ 60
sh ◦ 2 ◦ SLEEPING ◦ 60
foo ◦ 8 ◦ RUNNABLE ◦ 60
foo ◦ 7 ◦ SLEEPING ◦ 60
foo ◦ 6 ◦ SLEEPING ◦ 60
foo ◦ 9 ◦ RUNNING ◦ 5
ps ◦ 12 ◦ RUNNING ◦ 60
$ 7
```

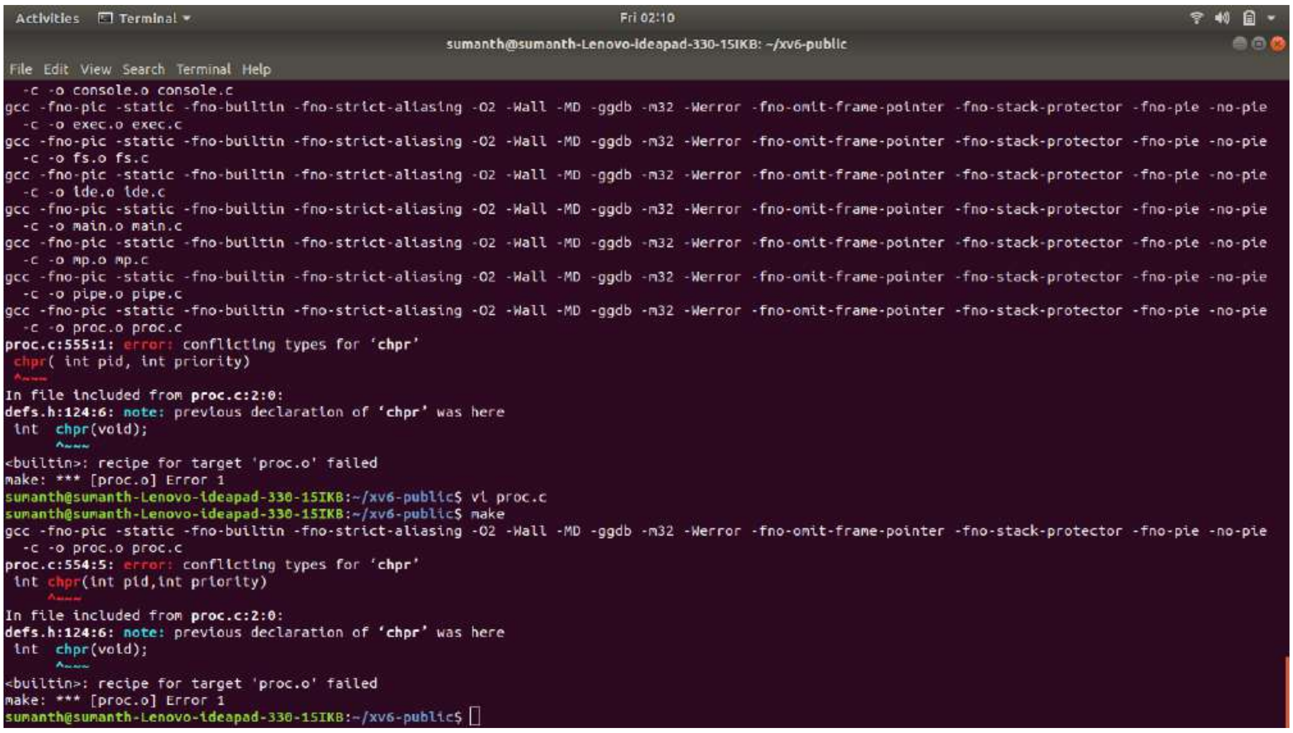
Issues faced (with relevant screenshots in all reviews):


Activities
Terminal
Thu 21:32
sumanth@sumanth-Lenovo-Ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

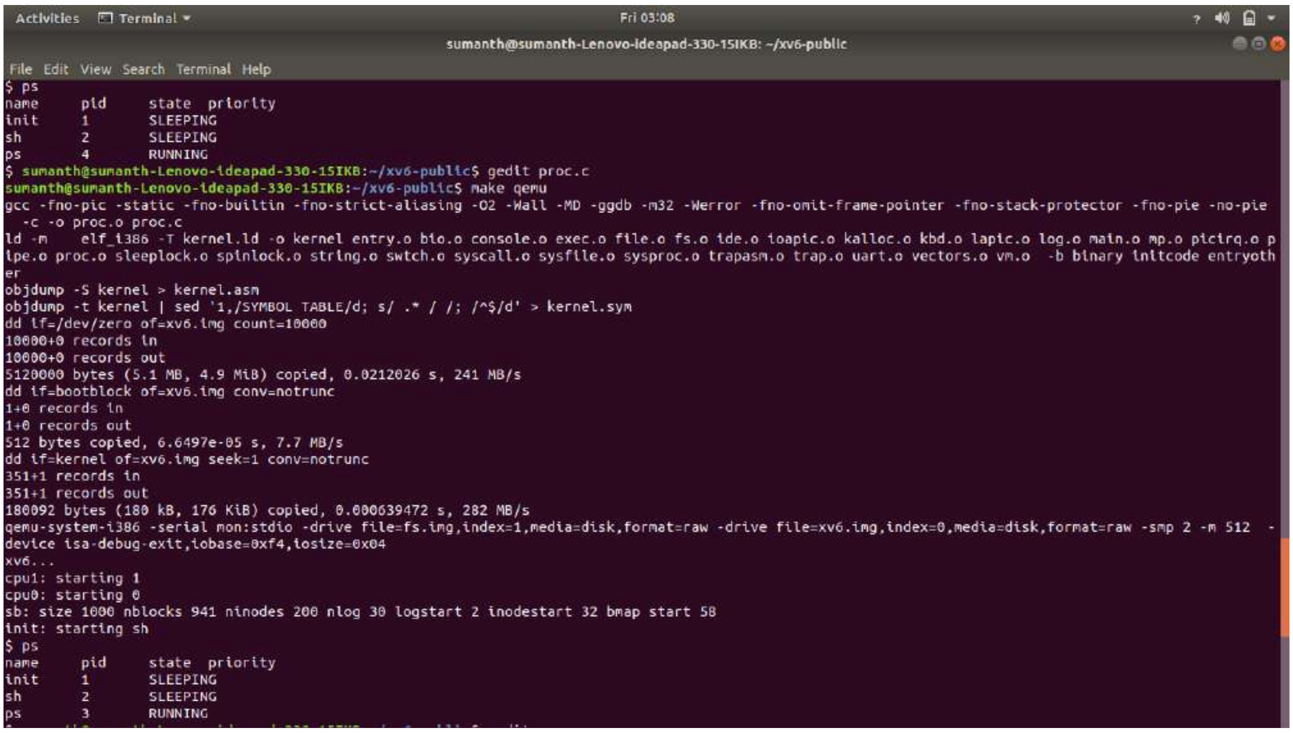
```
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~$ cd xv6-public
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ make qemu
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512  -
device isa-debug-exit,iobase=0xf4,iosize=0x04
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ vi user.h
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ make qemu
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c -o cat.o cat.c
cat.c: In function ‘sleep’:
cat.c:9:1: error: expected ‘=’, ‘,’, ‘;’, ‘asm’ or ‘__attribute__’ before ‘{’ token
 {
 ^
cat.c:26:1: error: expected ‘=’, ‘,’, ‘;’, ‘asm’ or ‘__attribute__’ before ‘{’ token
 {
 ^
In file included from cat.c:3:0:
user.h:24:5: error: old-style parameter declarations in prototyped function definition
 int sleep(int)
     ^~~~~
user.h:24:1: error: parameter name omitted
 int sleep(int)
 ^~~
cat.c:43:1: error: expected ‘{’ at end of input
 }
 ^
cat.c:43:1: error: control reaches end of non-void function [-Werror=return-type]
 }
 ^
cc1: all warnings being treated as errors
<builtin>: recipe for target 'cat.o' failed
make: *** [cat.o] Error 1
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ 
```


Activities
Terminal
Fri 02:00
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
objdump -t _parent | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > parent.sym
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c -o nice.o nice.c
nice.c:1:2: error: stray '#' in program
 i#include "types.h"
  ^
nice.c:1:1: error: unknown type name 'i'
 i#include "types.h"
 ^
nice.c:1:11: error: expected '=', ',', ';', 'asm' or '__attribute__' before string constant
 i#include "types.h"
           ^~~~~~~~~
In file included from nice.c:3:0:
user.h:38:1: error: unknown type name 'uint'; did you mean 'int'?
 uint strlen(const char*);
 ^~~~
 int
user.h:39:26: error: unknown type name 'uint'; did you mean 'int'?
 void* memset(void*, int, uint);
                          ^~~~
                          int
user.h:40:1: error: parameter names (without types) in function declaration [-Werror]
 void* malloc(uint);
 ^~~~
nice.c: In function 'main':
nice.c:24:5: error: too many arguments to function 'chpr'
     chpr(pid, priority);
     ^~~~
In file included from nice.c:3:0:
user.h:28:5: note: declared here
 int chpr(void);
     ^~~~
cc1: all warnings being treated as errors
<builtin>: recipe for target 'nice.o' failed
make: *** [nice.o] Error 1
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ vi nice,c
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ vi nice.c
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ make qemu
```


Activities
Terminal
Fri 02:10
sumanth@sumanth-Lenovo-ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
  -c -o console.o console.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c -o exec.o exec.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c -o fs.o fs.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c -o ide.o ide.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c -o main.o main.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c -o mp.o mp.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c -o pipe.o pipe.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c -o proc.o proc.c
proc.c:555:1: error: conflicting types for 'chpr'
 chpr( int pid, int priority)
 ^~~~
In file included from proc.c:2:0:
defs.h:124:6: note: previous declaration of 'chpr' was here
 int  chpr(void);
      ^~~~
<builtin>: recipe for target 'proc.o' failed
make: *** [proc.o] Error 1
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ vi proc.c
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ make
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c -o proc.o proc.c
proc.c:554:5: error: conflicting types for 'chpr'
 int chpr(int pid,int priority)
     ^~~~
In file included from proc.c:2:0:
defs.h:124:6: note: previous declaration of 'chpr' was here
 int  chpr(void);
      ^~~~
<builtin>: recipe for target 'proc.o' failed
make: *** [proc.o] Error 1
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ 
```


Activities
Terminal
Fri 03:08
sumanth@sumanth-Lenovo-Ideapad-330-15IKB: ~/xv6-public
File Edit View Search Terminal Help

```
$ ps
name     pid     state  priority
init     1       SLEEPING
sh       2       SLEEPING
ps       4       RUNNING
$ sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ gedit proc.c
sumanth@sumanth-Lenovo-ideapad-330-15IKB:~/xv6-public$ make qemu
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
  -c -o proc.o proc.c
ld -m    elf_i386 -T kernel.ld -o kernel entry.o bio.o console.o exec.o file.o fs.o ide.o ioapic.o kalloc.o kbd.o lapic.o log.o main.o mp.o picirq.o p
ipe.o proc.o sleeplock.o spinlock.o string.o swtch.o syscall.o sysfile.o sysproc.o trapasm.o trap.o uart.o vectors.o vm.o  -b binary initcode entryoth
er
objdump -S kernel > kernel.asm
objdump -t kernel | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > kernel.sym
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.0212026 s, 241 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 6.6497e-05 s, 7.7 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
351+1 records in
351+1 records out
180092 bytes (180 kB, 176 KiB) copied, 0.000639472 s, 282 MB/s
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512  -
device isa-debug-exit,iobase=0xf4,iosize=0x04
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ps
name     pid     state  priority
init     1       SLEEPING
sh       2       SLEEPING
ps       3       RUNNING
```

References:

1. https://www.youtube.com/watch?v=21SVYiKhcwM
2. https://medium.com/@silvamatteus/adding-new-system-calls-to-xv6-217b7daefbe1
3. https://medium.com/@viduniwickramarachchi/add-a-new-system-call-in-xv6-5486c2437573
4. https://stackoverflow.com/questions/8021774/how-do-i-add-a-system-call-utility-in-xv6
5. https://stackoverflow.com/questions/21653195/xv6-add-a-system-call-that-counts-system-calls
6. https://arjunkrishnababu96.gitlab.io/post/xv6-system-call/
7. https://github.com/sayak119/xv6_scheduler