Testing

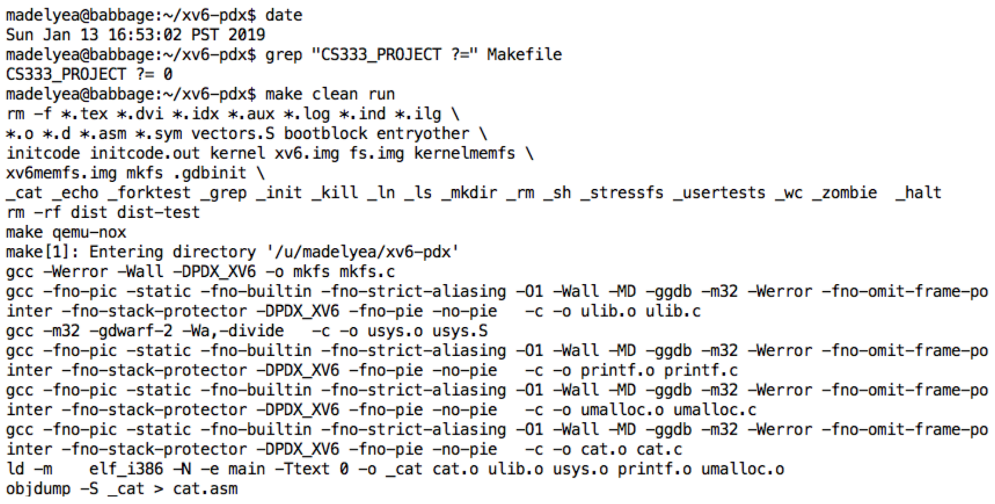
Compilation Test

This section will show that my project one code compiles with the project flag set to "1" and also set to "0".

Subtest 1: Compile with CS333_PROJECT set to 0. Since the listing is so long, this will require two screen shots.

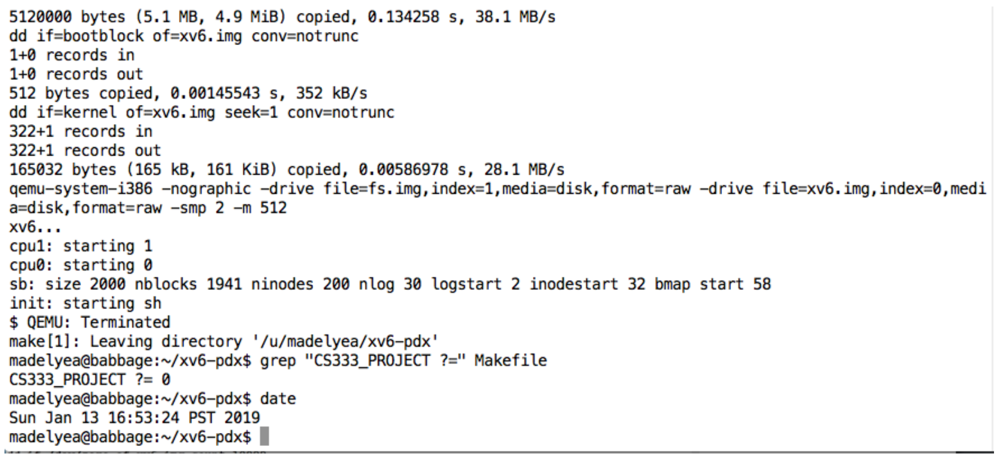
In the first screen shot, the current date and time is displayed as well as the value for CS333_PROJECT, verifying that it is set to 0. In the second screen shot, the same information is displayed. This is used to show that the two screen shots are from the same compilation sequence.

The expected outcome is that the compilation step will correctly compile with the project flag set to "0".



```
madelyea@babbage:~/xv6-pdx$ date
Sun Jan 13 16:53:02 PST 2019
madelyea@babbage:~/xv6-pdx$ grep "CS333_PROJECT ?=" Makefile
CS333_PROJECT ?= 0
madelyea@babbage:~/xv6-pdx$ make clean run
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 \
_cat _echo _forktest _grep _init _kill _ln _ls _mkdir _rm _sh _stressfs _usertests _wc _zombie  _halt
rm -rf dist dist-test
make qemu-nox
make[1]: Entering directory '/u/madelyea/xv6-pdx'
gcc -Werror -Wall -DPDX_XV6 -o mkfs mkfs.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O1 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-po
inter -fno-stack-protector -DPDX_XV6 -fno-pie -no-pie   -c -o ulib.o ulib.c
gcc -m32 -gdwarf-2 -Wa,-divide   -c -o usys.o usys.S
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O1 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-po
inter -fno-stack-protector -DPDX_XV6 -fno-pie -no-pie   -c -o printf.o printf.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O1 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-po
inter -fno-stack-protector -DPDX_XV6 -fno-pie -no-pie   -c -o umalloc.o umalloc.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O1 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-po
inter -fno-stack-protector -DPDX_XV6 -fno-pie -no-pie   -c -o cat.o cat.c
ld -m    elf_i386 -N -e main -Ttext 0 -o _cat cat.o ulib.o usys.o printf.o umalloc.o
objdump -S _cat > cat.asm
```

Figure 1: Compilation with CS333_PROJECT set to 0.



```
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.134258 s, 38.1 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.00145543 s, 352 kB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
322+1 records in
322+1 records out
165032 bytes (165 kB, 161 KiB) copied, 0.00586978 s, 28.1 MB/s
qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,medi
a=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 2000 nblocks 1941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ QEMU: Terminated
make[1]: Leaving directory '/u/madelyea/xv6-pdx'
madelyea@babbage:~/xv6-pdx$ grep "CS333_PROJECT ?=" Makefile
CS333_PROJECT ?= 0
madelyea@babbage:~/xv6-pdx$ date
Sun Jan 13 16:53:24 PST 2019
madelyea@babbage:~/xv6-pdx$
```

Figure 2: Compilation with CS333_PROJECT set to 0.

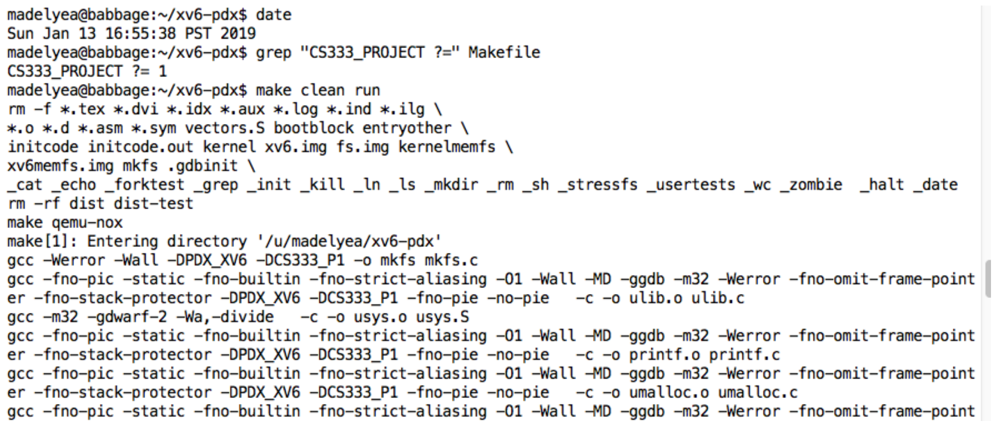
The date in the first and second figures show about 22 seconds of elapsed time. This shows that the two date commands occurred close in time. The grep commands before and after the compilation show that the project flag in the Makefile is set to “0”. The date commands are executed close in time, the project flag shows the same value before and after the compilation, and the compilation shows no errors. This leads to the conclusion that the project code correctly compiles with the project flag turned off.

This subtest **PASSES**.

Subtest 2: Boot compile with CS333_PROJECT set to 1. Since the listing is so long, this will require two screen shots.

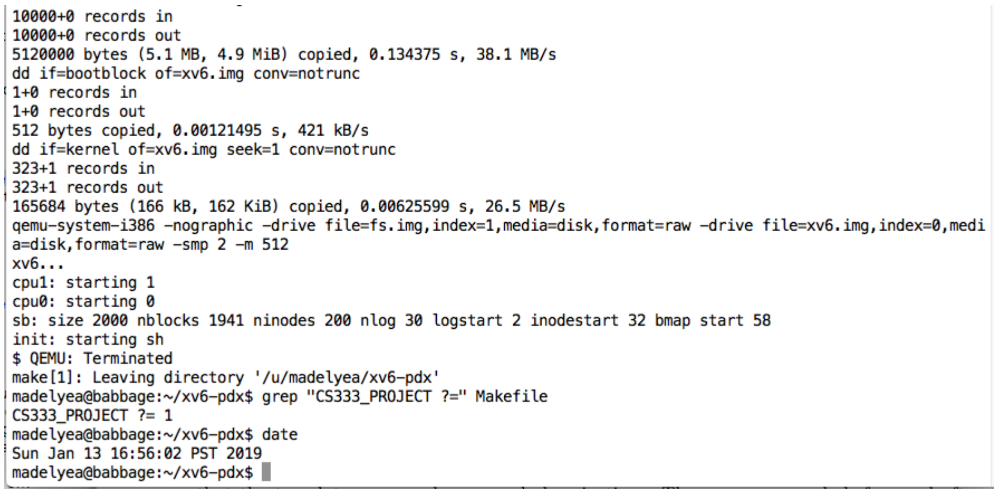
In the first screen shot, the current date and time is displayed as well as the value for CS333_PROJECT, verifying that it is set to 1. In the second screen shot, the same information is displayed. This is used to show that the two screen shots are from the same compilation sequence.

The expected outcome is that the compilation step will correctly compile with the project flag set to “1”.



```
madelyea@babbage:~/xv6-pdx$ date
Sun Jan 13 16:55:38 PST 2019
madelyea@babbage:~/xv6-pdx$ grep "CS333_PROJECT ?=" Makefile
CS333_PROJECT ?= 1
madelyea@babbage:~/xv6-pdx$ make clean run
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 \
_cat _echo _forktest _grep _init _kill _ln _ls _mkdir _rm _sh _stressfs _usertests _wc _zombie  _halt _date
rm -rf dist dist-test
make qemu-nox
make[1]: Entering directory '/u/madelyea/xv6-pdx'
gcc -Werror -Wall -DPDX_XV6 -DCS333_P1 -o mkfs mkfs.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O1 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-point
er -fno-stack-protector -DPDX_XV6 -DCS333_P1 -fno-pie -no-pie   -c -o ulib.o ulib.c
gcc -m32 -gdwarf-2 -Wa,-divide   -c -o usys.o usys.S
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O1 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-point
er -fno-stack-protector -DPDX_XV6 -DCS333_P1 -fno-pie -no-pie   -c -o printf.o printf.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O1 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-point
er -fno-stack-protector -DPDX_XV6 -DCS333_P1 -fno-pie -no-pie   -c -o umalloc.o umalloc.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O1 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-point
```

Figure 3: Compilation with CS333_PROJECT set to 1.



```
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.134375 s, 38.1 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.00121495 s, 421 kB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
323+1 records in
323+1 records out
165684 bytes (166 kB, 162 KiB) copied, 0.00625599 s, 26.5 MB/s
qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,medi
a=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 2000 nblocks 1941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ QEMU: Terminated
make[1]: Leaving directory '/u/madelyea/xv6-pdx'
madelyea@babbage:~/xv6-pdx$ grep "CS333_PROJECT ?=" Makefile
CS333_PROJECT ?= 1
madelyea@babbage:~/xv6-pdx$ date
Sun Jan 13 16:56:02 PST 2019
madelyea@babbage:~/xv6-pdx$
```

Figure 4: Compilation with CS333_PROJECT set to 1.

The date in the first and second figures show about 24 seconds of elapsed time. This shows that the two date commands occurred close in time. The grep commands before and after the compilation show that the project flag in the Makefile is set to “1”. The date commands are executed close in time, the project flag shows the same value before and after the compilation, and the compilation shows no errors. This leads to the conclusion that the project code correctly compiles with the project flag turned off.

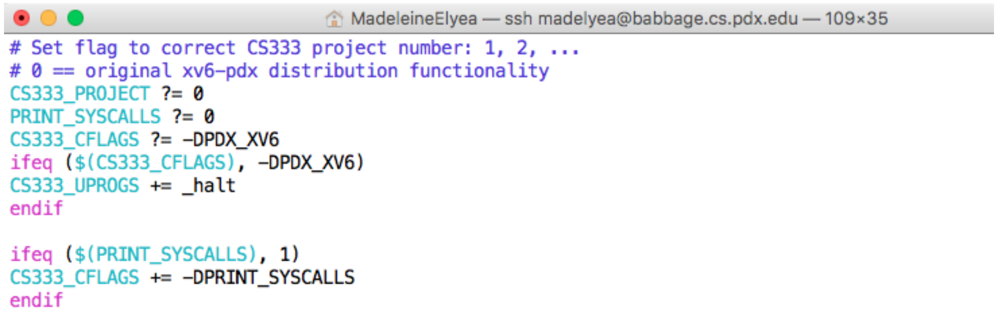
This subtest **PASSES**.

Since each subtest passes and the subtests fully test the objectives, this test **PASSES**.

PRINT_SYSCALLS Test

This test verifies that my kernel correctly compiles with the flag `PRINT_SYSCALLS` turned off, set to 0 in the Makefile. This test has two subtests: 1) the kernel correctly compiles and boots with the `CS333_P1`— flag turned off; and 2) the kernel correctly compiles and boots with the `CS333_P1`— flag turned on. Since the `PRINT_SYSCALLS` causes all system calls to be printed along with their return codes, booting to the shell is sufficient as that process causes many system calls to be invoked. It is expected that no system call information will be printed to the console.

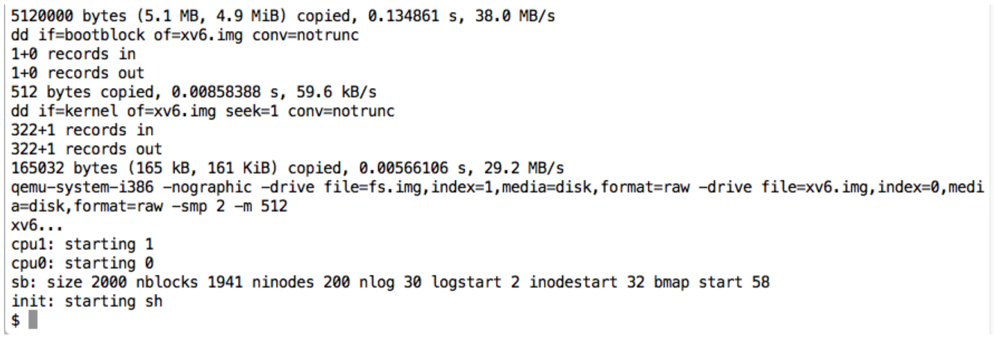
Subtest 1: PRINT_SYSCALLS and CS333_PROJECT set to 0.



```
# Set flag to correct CS333 project number: 1, 2, ...
# 0 == original xv6-pdx distribution functionality
CS333_PROJECT ?= 0
PRINT_SYSCALLS ?= 0
CS333_CFLAGS ?= -DPDX_XV6
ifeq ($(CS333_CFLAGS), -DPDX_XV6)
CS333_UPROGS += _halt
endif

ifeq ($(PRINT_SYSCALLS), 1)
CS333_CFLAGS += -DPRINT_SYSCALLS
endif
```

Figure 5: Makefile with PRINT_SYSCALLS and CS333_PROJECT set to 0.

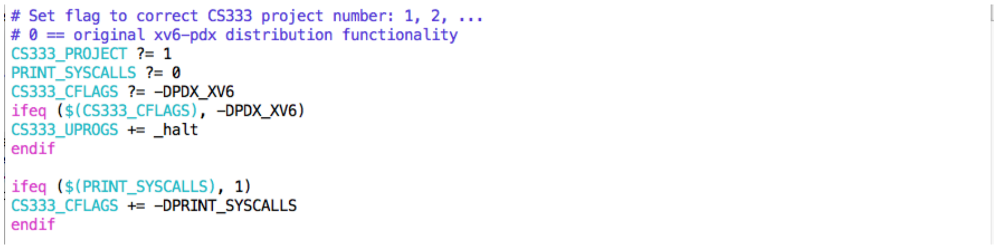


```
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.134861 s, 38.0 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.00858388 s, 59.6 kB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
322+1 records in
322+1 records out
165032 bytes (165 kB, 161 KiB) copied, 0.00566106 s, 29.2 MB/s
qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,medi
a=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 2000 nblocks 1941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ 
```

Figure 6: Boot with PRINT_SYSCALLS and CS333_PROJECT set to 0.

No system call information is displayed on boot with PRINT_SYSCALLS and CS333_PROJECT set to 0, as expected. This subtest **PASSES**.

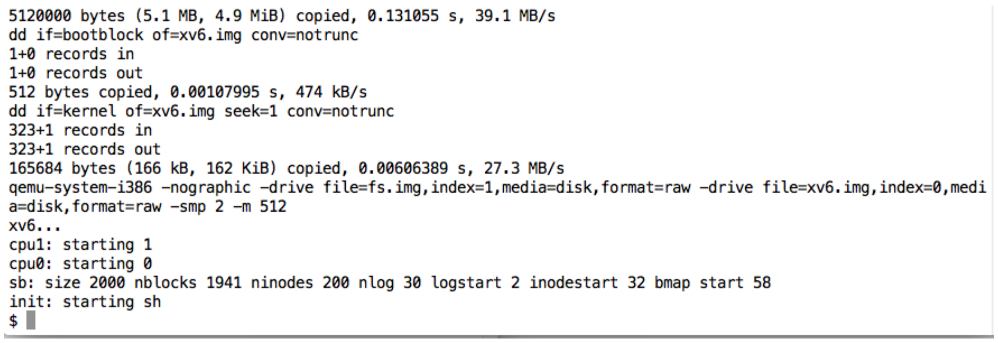
Subtest 2: PRINT_SYSCALLS set to 0 and CS333_PROJECT set to 1.



```
# Set flag to correct CS333 project number: 1, 2, ...
# 0 == original xv6-pdx distribution functionality
CS333_PROJECT ?= 1
PRINT_SYSCALLS ?= 0
CS333_CFLAGS ?= -DPDX_XV6
ifeq ($(CS333_CFLAGS), -DPDX_XV6)
CS333_UPROGS += _halt
endif

ifeq ($(PRINT_SYSCALLS), 1)
CS333_CFLAGS += -DPRINT_SYSCALLS
endif
```

Figure 7: Makefile with PRINT_SYSCALLS set to 0 and CS333_PROJECT set to 1.



```
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.131055 s, 39.1 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.00107995 s, 474 kB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
323+1 records in
323+1 records out
165684 bytes (166 kB, 162 KiB) copied, 0.00606389 s, 27.3 MB/s
qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,medi
a=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 2000 nblocks 1941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$
```

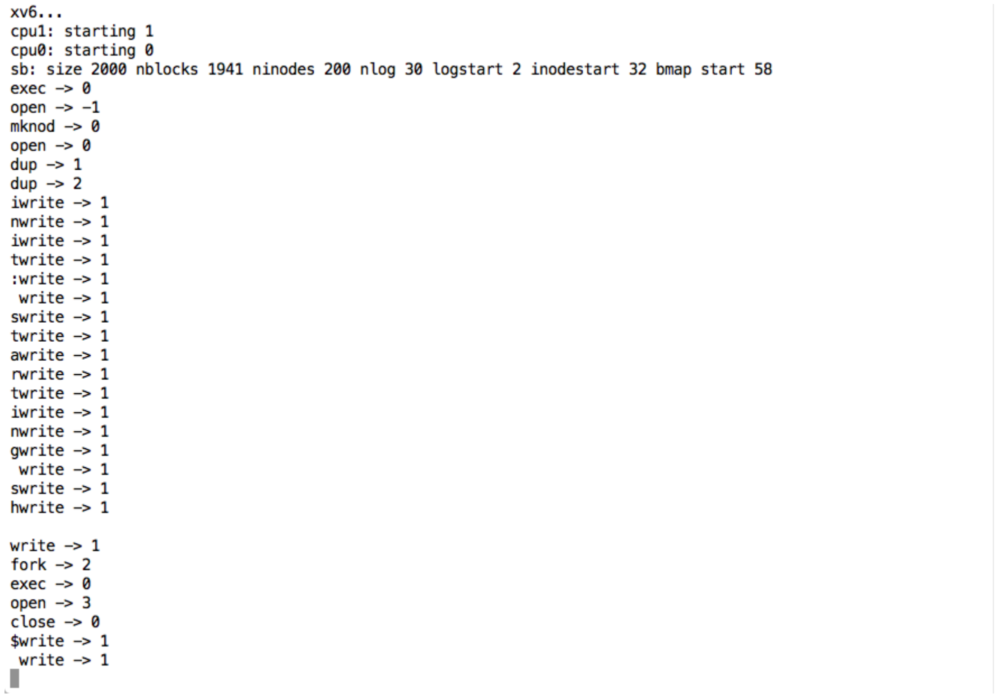
Figure 8: Boot with PRINT_SYSCALLS set to 0 and CS333_PROJECT set to 1.

No system call information is displayed on boot with PRINT_SYSCALLS set to 0 and CS333_PROJECT set to 1, as expected. This subtest **PASSES**.

Both subtests pass. This test therefore **PASSES**.

System Call Tracing Facility

This test verifies that my kernel correctly compiles with the flag `PRINT_SYSCALLS` turned on, set to 1 in the Makefile. This test boots the kernel to the shell prompt. The output should contain additional information from the `PRINT_SYSCALLS turned off` test; specifically a list of system calls and their return codes should be displayed. This list should closely match the output shown in the project description.



```
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 2000 nblocks 1941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
exec -> 0
open -> -1
mknod -> 0
open -> 0
dup -> 1
dup -> 2
iwrite -> 1
nwrite -> 1
iwrite -> 1
twrite -> 1
:write -> 1
 write -> 1
swrite -> 1
twrite -> 1
awrite -> 1
rwrite -> 1
twrite -> 1
iwrite -> 1
nwrite -> 1
gwrite -> 1
 write -> 1
swrite -> 1
hwrite -> 1

write -> 1
fork -> 2
exec -> 0
open -> 3
close -> 0
$write -> 1
 write -> 1
```

Figure 9: Boot with PRINT_SYSCALLS set to 1 and CS333_PROJECT set to 1.

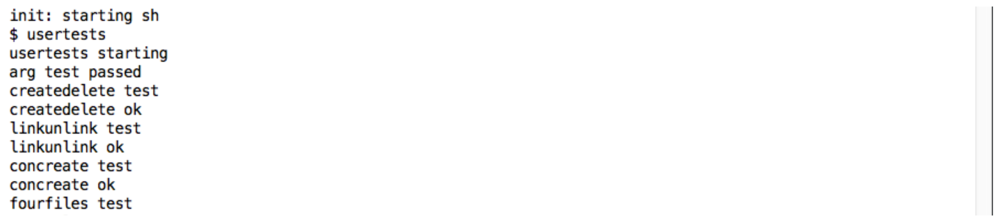
The system call trace correctly displays the invoked system calls and matches the reference output from the project description. Standard output is interleaved with the trace output, as expected.

This test **PASSES**.

Usertests and forktest run with CS333_P1 flag turned off

I tested that xv6 correctly compiles and runs with the `CS333_P1` flag disabled. I set the `CS333_PROJECT` value in the `Makefile` to 0. I then verified this setting and compiled xv6.

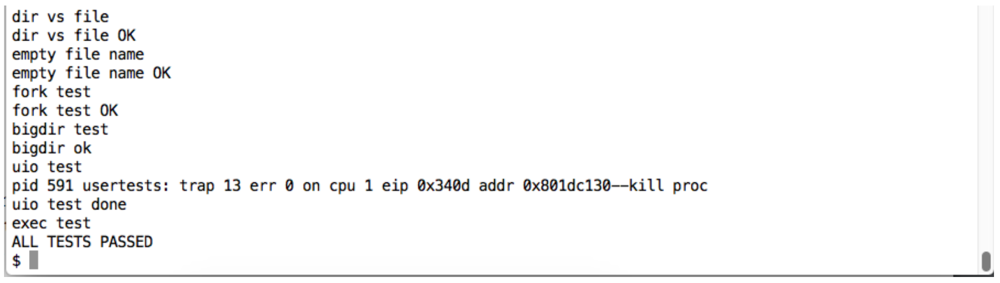
It is expected that xv6 will boot normally and both `usertests` and `forktest` will successfully execute. Since `forktest` is executed as part of `usertests`, only `usertests` will be executed.



```
init: starting sh
$ usertests
usertests starting
arg test passed
createdelete test
createdelete ok
linkunlink test
linkunlink ok
concreate test
concreate ok
fourfiles test
```

Figure 10: CS333_P1 disabled 1

Some output has been elided.



```
dir vs file
dir vs file OK
empty file name
empty file name OK
fork test
fork test OK
bigdir test
bigdir ok
uio test
pid 591 usertests: trap 13 err 0 on cpu 1 eip 0x340d addr 0x801dc130--kill proc
uio test done
exec test
ALL TESTS PASSED
$ 
```

Figure 11: CS333_P1 disabled 2

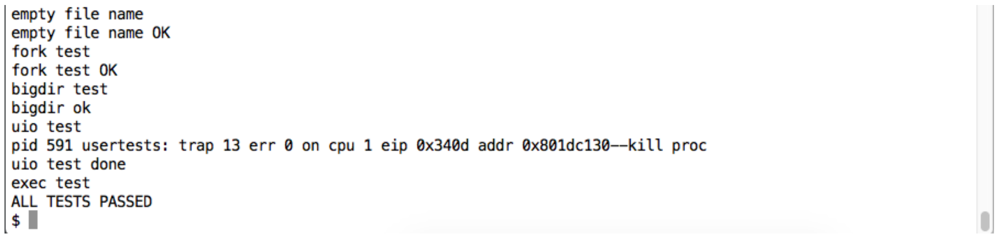
From both figures, we can see that all usertests have passed. Further, since `forktest` is run as a part of `usertests`, we know that `forktest` has passed.

This test **PASSES**.

Usertests and forktest run with CS333_P1 flag turned on

I tested that xv6 correctly compiles and runs with the `CS333_P1` flag enabled. I set the `CS333_PROJECT` value in the `Makefile` to 1, compiled and booted xv6 using `make qemu-nox`, and then ran `usertests`. As mentioned above, this is an acceptable test for both `usertests` and `forktest` since `forktest` is run as part of `usertests`.

It is expected that all tests from `usertests` will pass.



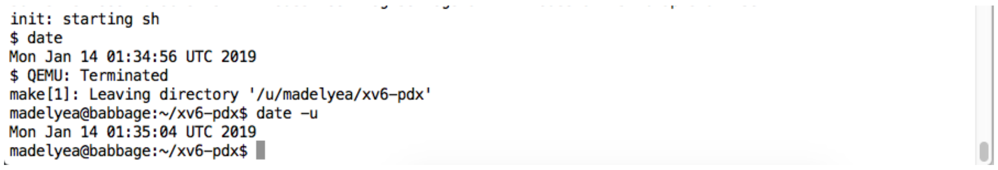
```
empty file name
empty file name OK
fork test
fork test OK
bigdir test
bigdir ok
uio test
pid 591 usertests: trap 13 err 0 on cpu 1 eip 0x340d addr 0x801dc130--kill proc
uio test done
exec test
ALL TESTS PASSED
$ 
```

Figure 12: Usertests with CS333_P1 enabled

From the above figure, we can see that all usertest tests pass. This test **PASSES**.

Date Command

This test verifies that my date command works correctly. As implementing timezones was not a requirement and the system clock tracks to universal coordinate time (UTC), the expected output would be a close match to running the Linux date command `date -u`. The test will require 1) boot xv6; 2) run the date command under xv6; 3) exit xv6 (I will use the control sequence); and 4) run "date -u" at the Linux prompt. Note that the Linux output is expected to display a few seconds later than the xv6 date command as it takes non-zero time to perform the xv6 shutdown and Linux command invocation.



```
init: starting sh
$ date
Mon Jan 14 01:34:56 UTC 2019
$ QEMU: Terminated
make[1]: Leaving directory '/u/madelyea/xv6-pdx'
madelyea@babbage:~/xv6-pdx$ date -u
Mon Jan 14 01:35:04 UTC 2019
madelyea@babbage:~/xv6-pdx$
```

Figure 13: Date Command

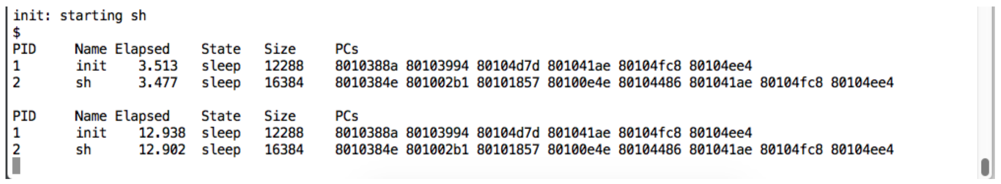
As expected, the xv6 date command prints the same information in the same format as the Linux date command and the seconds field is a few seconds later for the Linux command than the xv6 command.

This test **PASSES**.

Control-P Format

In this test, I verified that Control-P displays processes with the correct header, that process information is aligned with the appropriate header, and that the correct data is displayed.

It is expected that I will observe a well-formatted and correct output from the Control-P command.



```
init: starting sh
$
PID     Name Elapsed    State   Size     PCs
1       init    3.513   sleep   12288    8010388a 80103994 80104d7d 801041ae 80104fc8 80104ee4
2       sh      3.477   sleep   16384    8010384e 801002b1 80101857 80100e4e 80104486 801041ae 80104fc8 80104ee4

PID     Name Elapsed    State   Size     PCs
1       init    12.938  sleep   12288    8010388a 80103994 80104d7d 801041ae 80104fc8 80104ee4
2       sh      12.902  sleep   16384    8010384e 801002b1 80101857 80100e4e 80104486 801041ae 80104fc8 80104ee4
```

Figure 14: Control-p output

From the above figure, we can see that the header contains the appropriate fields, that the appropriate process information is aligned with each header item, and that the process information displayed is correct (it is assumed that the program counters are correct).

The elapsed time for the `init` process is slightly higher than for the `sh` process. This is expected since the `init` process is the first process to start and `init` is responsible for starting `sh`.

In addition, the elapsed time increases with ever press of Control-P. This is the expected and desired behavior.

This test **PASSES**.