Project 1 Test Report

Alexander DuPree

January 9, 2020

Introduction

The following test report documents the tests performed for project one. The test cases and strategies closely follow the project one rubric.

Each section contains test cases related to the sections topic. Each test case will describe the name of the test, the expected result, actual result, as well as a discussion and indication of the Pass/Fail status. The actual result will be provided in the form of a screen shot of the console.

System Call Tracing

This section presents all tests related to tracing system calls within the xv6 system. Test cases follow closely those outlined in the rubric.

Test Case: *With PRINT_SYSCALLS set to 0 in the Makefile*
Assertions:

1. Code correctly compiles
2. No trace information is displayed

Status: **PASS**

```
|20:01:33|adupree@ada:[xv6-pdx]> grep "PRINT_SYSCALLS ?=" Makefile
PRINT_SYSCALLS ?=0
|20:02:01|adupree@ada:[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 -s clean
make -s qemu-nox
nmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks 1941 total 2000
balloc: first 582 blocks have been allocated
balloc: write bitmap block at sector 58
boot block is 448 bytes (max 510)
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.160216 s, 32.0 MB/s
1+0 records in
1+0 records out
512 bytes copied, 0.00954813 s, 53.6 kB/s
315+1 records in
315+1 records out
161360 bytes (161 kB, 158 KiB) copied, 0.017853 s, 9.0 MB/s
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 1: Compilation and execution with PRINT_SYSCALLS set to 0.

The command `grep "PRINT_SYSCALLS ?=" Makefile` shows that the PRINT_SYSCALLS macro is indeed turned off. The following command `make clean run` demonstrates that the code correctly compiles and the output does not display any trace information. Furthermore, it should be noted that the timestamps for each command was performed within thirty seconds of each other.

Test Case: *With PRINT_SYSCALLS set to 1 in the Makefile*
Assertions:

1. Code correctly compiles
2. Trace information is displayed
3. Correct call trace on boot

Status: **PASS**

```
|20:09:07|adupree@ada:[xv6-pdx]> grep "PRINT_SYSCALLS ?=" Makefile
PRINT_SYSCALLS ?=1
|21:15:15|adupree@ada:[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 -s clean
make -s qemu-nox
nmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks 1941 total 2000
balloc: first 582 blocks have been allocated
balloc: write bitmap block at sector 58
boot block is 448 bytes (max 510)
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.161945 s, 31.6 MB/s
1+0 records in
1+0 records out
512 bytes copied, 0.0105074 s, 48.7 kB/s
315+1 records in
315+1 records out
161472 bytes (161 kB, 158 KiB) copied, 0.0155575 s, 10.4 MB/s
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 2: Compilation and Boot call trace with PRINT_SYSCALLS set to 1.

The command `grep "PRINT_SYSCALLS ?=" Makefile` shows that the PRINT_SYSCALLS macro is indeed turned on. The following command `make clean run` demonstrates that the code correctly compiles and the output displays trace information for the system calls. The system call trace in Figure 2 matches the trace in the project description, which means that this is the correct call trace on boot. Furthermore, it should be noted that the timestamps for each command was performed within 10 seconds of each other.

Conditional Compilation and Date System Call

This section presents tests related to the conditional compilation of the CS333_P1 project flag and the Date system call. Test cases follow closely those outlined in the rubric.

Test Case: *With CS333_P1 turned off, when CS333_PROJECT is set to 0*
Assertions:

1. Code correctly compiles
2. Kernel boots
3. forktest runs to completion correctly
4. usertests runs to completion correctly

Status: **PASS**

```
|21:44:11|adupree@ada:[xv6-pdx]> grep "CS333_PROJECT ?=" Makefile
CS333_PROJECT ?= 0
|21:44:17|adupree@ada:[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 -s clean
make -s qemu-nox
nmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks 1941 total 2000
balloc: first 582 blocks have been allocated
balloc: write bitmap block at sector 58
boot block is 448 bytes (max 510)
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.150241 s, 34.1 MB/s
1+0 records in
1+0 records out
512 bytes copied, 0.0100407 s, 51.0 kB/s
315+1 records in
315+1 records out
161360 bytes (161 kB, 158 KiB) copied, 0.0149619 s, 10.8 MB/s
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 3: Compilation and Kernel Boot with CS333_P1 turned off.

Figure 3, 4, and 5 present the results of this test case. The command `grep "CS333_PROJECT ?=" Makefile` shows that the CS333_PROJECT is defined with 0. The following command `make clean run` demonstrates that the code correctly compiles the kernel successfully boots. Because `forktest` is included with the `usertests` only the user tests were ran. The results of which are shown in figure 4 and 5. Due to the size of the test, much of the output was elided.

```
|21:44:17|adupree@ada:[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 -s clean
make -s qemu-nox
nmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks 1941 total 2000
balloc: first 582 blocks have been allocated
balloc: write bitmap block at sector 58
boot block is 448 bytes (max 510)
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.150241 s, 34.1 MB/s
1+0 records in
1+0 records out
512 bytes copied, 0.0100407 s, 51.0 kB/s
315+1 records in
315+1 records out
161360 bytes (161 kB, 158 KiB) copied, 0.0149619 s, 10.8 MB/s
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
$ usertests
usertests starting
arg test passed
```

Figure 4: Execution of usertests from the same session.

```
unlinkread test
unlinkread ok
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 0x33ce addr 0x8013d620--kill proc
uio test done
exec test
ALL TESTS PASSED
$
```

Figure 5: Results of usertests with output elided.

Test Case: *With CS333_P1 turned on, when CS333_PROJECT is set to 1*
Assertions:

1. Code correctly compiles
2. Kernel boots
3. forktest runs to completion correctly
4. usertests runs to completion correctly
5. The date command prints correct information
6. The date command prints information in the correct format

Status: PASS

```
|09:37:23|adupree@ada:[xv6-pdx]> grep "CS333_PROJECT ?=" Makefile
CS333_PROJECT ?= 1
|09:37:34|adupree@ada:[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 -s clean
make -s qemu-nox
nmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks 1941 total 2000
balloc: first 611 blocks have been allocated
balloc: write bitmap block at sector 58
boot block is 448 bytes (max 510)
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.163607 s, 31.3 MB/s
1+0 records in
1+0 records out
512 bytes copied, 0.00505249 s, 101 kB/s
316+1 records in
316+1 records out
161856 bytes (162 kB, 158 KiB) copied, 0.0143233 s, 11.3 MB/s
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: Compilation and Kernel Boot with CS333_P1 turned on.

```
|09:37:34|adupree@ada:[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 -s clean
make -s qemu-nox
nmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks 1941 total 2000
balloc: first 611 blocks have been allocated
balloc: write bitmap block at sector 58
boot block is 448 bytes (max 510)
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.163607 s, 31.3 MB/s
1+0 records in
1+0 records out
512 bytes copied, 0.00505249 s, 101 kB/s
316+1 records in
316+1 records out
161856 bytes (162 kB, 158 KiB) copied, 0.0143233 s, 11.3 MB/s
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
$ usertests
usertests starting
arg test passed
```

Figure 7: Execution of usertests from the same session with CS333_P1 defined.

```
unlinkread test
unlinkread ok
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 0x33ce addr 0x801dc130--kill proc
uio test done
exec test
ALL TESTS PASSED
$
```

Figure 8: Results of usertests with output elided and CS333_P1 defined.

```
ALL TESTS PASSED
$ date -u
Thu Jan 9 17:42:45 UTC 2020
$ date -u
Thu Jan 9 17:42:52 UTC 2020
$ halt

Shutting down ...
reset -I
|09:42:56|adupree@ada:[xv6-pdx]> date -u
Thu Jan  9 17:42:58 UTC 2020
|09:42:58|adupree@ada:[xv6-pdx]>
```

Figure 9: Execution of date command, twice within xv6 and once on host machine.

Figures 6, 7, 8, and 9 present the results of this test case. In figure 6, the command 'grep "CS333_PROJECT ?= Makefile' shows the the CS333_PROJECT flag is set to 1. Following that is the command `make clean run` which demonstrates that the code correctly compiles and the kernel boots successfully. Because `forktest` is included with the `usertests`, only the user tests were ran, as illustrated in Figure 7. Figure 8 shows that all user tests pass. Figure 9 presents that the date system call is working correctly and displays the correct information in the correct format. Note that the command `date -u` is issued twice within the xv6 session to demonstrate that the date command provides different timestamps as expected. Also, the `date -u` is issued in the host machines terminal as well to prove that the inormation and format is correct. Furthermore, the timestamps from each date call is different and are done over a span of 15 seconds.

Control-P

This section presents tests related to the Control-P console interrupt and process information dump. Test cases follow closely those outlined in the rubric.

Test Case: *Issuing control sequence 'Control-P' at runtime*
Assertions:

1. Correct header displayed
2. Output is aligned with header
3. Correct and updated data is displayed

Status: **PASS**

```
|21:28:50|adupree@babbage:[xv6-pdx]> make qemu-nox
qemu-system-i386 -nographic -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 2000 nblocks 1941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$
PID     Name         Elapsed    State   Size     PCs
1       init         4.490s     sleep   12288    80103755 80103864 80104c32 80104023 80104e8d 80104d83
2       sh           4.464s     sleep   16384    80103755 801002c4 80101814 80100e40 80104356 80104023 80104e8d 80104d83
$
PID     Name         Elapsed    State   Size     PCs
1       init         9.12s      sleep   12288    80103755 80103864 80104c32 80104023 80104e8d 80104d83
2       sh           8.986s     sleep   16384    80103755 801002c4 80101814 80100e40 80104356 80104023 80104e8d 80104d83
$ date -u
Fri Jan 10 05:29:07 UTC 2020
$
```

Figure 10: Compilation and Boot call trace with PRINT_SYSCALLS set to 1.

First, we can see that the header presented in Figure 10 displays the required fields outlined in the project description. The data displayed is aligned with the correct fields and is correct for the system. As expected, the init process is the oldest process, and succesive uses of 'control-p' shows the elapsed time to be increasing for all processes.