Mark Gameng

CS 450 – Duan Yue

## Lab 2

### exit() to exitStatus(int status)

Rather than just changing the existing exit system call and thereby, update all the code that used exit(), I just created a new exit call, named exitStatus(int status).

For proc.c, It’s the same code as the original exit() but I just saved the exit status to the curproc. This saves the exit status of the current process.

Text

Description automatically generated

For proc.h, I then added int status in the struct.

Text

Description automatically generated

I then declared exitStatus in user level in user.h



I also updated usys.S to have exitStatus.



In syscall.c, I then added sys\_exitStatus.



Text

Description automatically generated

Same thing in syscall.h



In sysproc.c, I just used a similar structure as sys\_exit() and passed in an integer, status.

Text

Description automatically generated

I then defined it in defs.h



Now, exitStatus should be all implemented and what’s left is testing …

### wait() -> int wait(int \*status) and adding int waitpid(int pid, int \*status, int options)

Now, for updating wait and adding waitpid, its very similar to the process I did previously. Similar to exit, I made a new system call rather than updating wait, as I would have to update all the other instances of wait in xv6, which isn’t that much compared to exit, but still.

For wait, I added int waitStatus, which was just the same code as the original wait, but I added code to pass back the status.

Text

Description automatically generated

For waitpid, its similar to wait, but waits for a process with the given pid. Also, must wait for any process, meaning it doesn’t have to be a child process.

Text

Description automatically generated

I also modified sysproc.c for waitStatus and waitpid.

Text

Description automatically generated

Similar to exit, for waitStatus and waitpid, I modified user.h, usys.S, syscall.c, syscall.h, defs.h

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

Text

Description automatically generated

### Testing

For testing, I first used the given usertests by xv6. Running it gave no errors and outputs “ALL TESTS PASSED”, so I then started to make my own tests for the above functions, using test.c

To be able to run the test file, I needed to update the makefile.

A picture containing calendar

Description automatically generated

For test.c, this is my code.

Text

Description automatically generated

Text

Description automatically generated

Running these tests in xv6, results in:

Text

Description automatically generated

Text

Description automatically generated

Aside from overlap from print statements, due to running at the same time, the outputs are correct and show that the methods exitStatus(int status), waitStatus(int \*status), and waitpid(int pid, int \*status, int options) I implemented are working.

In the picture above, the parent is waiting for child (pid = 13) to exit and doesn’t run any code until it does. Once that the pid = 13 exits, can see it print above, overlapping with the parent, then the parent resumes execution and prints out that the child exited and then proceeds to wait for the next child.

I also ran diff -r original\_xv6 lab2\_xv6 to show all my code changes. That file, lab2diff.txt, is included in the submission.