Lab 1 Design Document

Khuaja Shams
ksham001@ucr.edu
Section 21

Alexander Nguyen
anguy086@ucr.edu
Section 21

Updates:

Changed files:
Proc.h
proc.c
user.h
defs.h
sysproc.c
syscall.c
syscall.h
usys.S
Makefile

Added Files:
mytest.c

proc.h:
We added a status field for exit, wait and waitpid

proc.c:
We added additional functionalities for exit and wait relating to the status field. We then created the waitpid function.

user.h and defs.h:
We added the exit, wait and waitpid functions.

sysproc.c:
We used the functions argint() and argptr() to update the functionalities of sys_exit and sys_wait to store the status of the process. We also created a sys_waitpid function.

syscall.c, syscall.h, and usys.S:
We added sys_waitpid here

Makefile:
We created and added mytest.c (test harness) to the Makefile so we can check our waitpid function to verify that it works correctly.

Adding a New File:

Suppose we have a valid hello world program named helloworld.c, for example. In order to execute it in the xv6 shell, we will first have to update the Makefile to include the program. In the UPROGS section around line 160, we would add:

_helloworld\

Then in the EXTRA section around line 240, we will add

helloworld.c

Once we have that, we can start gdb and create a breakpoint at main and continue to that line. Then continue again after that to start the xv6 shell. Then, we will type ./helloworld to execute the program.

Implementing Wait:

When the process has no child, sys_wait will return -1, indicating an error. When it has a child process, it will return it's child's process ID.

Testing exit, wait, and waitpid with given test harness:

The child's exit status was consistent with the parent's predicted exit status.

Testing Exit and Wait:

```
$ ./mytest 1

 This program tests the correctness of your lab#1

  Step 1: testing exit(int status) and wait(int* status):

This is child with PID# 4 and I will exit with status 0

 This is the parent: child with PID# 4 has exited with status 0

This is child with PID# 5 and I will exit with status -1

 This is the parent: child with PID# 5 has exited with status -1
$
```

Testing waitpid:

```
$ ./mytest 2

 This program tests the correctness of your lab#1

  Step 2: testing waitpid(int pid, int* status, int options):


  The is child with PID# 8 and I will exit with status 0

 The is child with PID# 10 and I will exit with status 0
The is child
 The is child with PID# 9 and I will exit with status 0
with PID# 7 and I will exit with status 0

 Th
 Thie is child with PID# 11 and I will exit with status 0
s is the parent: Now waiting for child with PID# 10

 This is the partent: Child# 7 has exited with status 0

 This is the parent: Now waiting for child with PID# 8

 This is the partent: Child# 9 has exited with status 0

 This is the parent: Now waiting for child with PID# 9

 This is the partent: Child# 8 has exited with status 0

 This is the parent: Now waiting for child with PID# 7

 This is the partent: Child# 10 has exited with status 0

 This is the parent: Now waiting for child with PID# 11
```