Daniel Stinson-Diess
Kennen DeRenard

Lab 1 Design Document & Report

Report:

Files Changed:

1. proc.h: added int ExitStatus
2. proc.c: add waitpid w/ implementation, edit wait/exit to add parameters.
3. defs.h: add int parameter to exit, and int* parameter to wait, create waitpid(int, int*, int*)
4. sysproc.c: add argint(...)/argptr(...) parameter fetches to exit/wait system calls. Add waitpid wrapper function.
5. syscall.c: add waitpid to trap vector table
6. syscall.h: add trap vector number to waitpid
7. user.h: update function signatures for all system calls
8. usys.S: add waitpid to syscall list
9. waitpidtest.c: create test harness for waitpid
10. Makefile: add userland tests under the UPROGS target.
11. Edit all previous instances of exit/wait to take in the correct function signatures.

Adding Hello World System Call:

1. In usys.S:
    a. Add “SYSCALL(hello)”
2. In syscall.h:
    a. Add “#define SYS_hello ##”
3. In syscall.c:
    a. Add “extern int sys_hello(void);”
    b. Add [SYS_hello] sys_hello,”
4. In sysproc.c:
    a. Add wrapper function int sys_hello(void)
5. In proc.c:
    a. Add implementation of hello system call
6. In defs.h:
    a. Add “void hello(void);” to section for proc.c
7. In user.h:
    a. Add “int hello(void):” to system calls section
8. In test.c:

    a. Create file which is the test harness for the system call.
9. In Makefile:
    a. Add _test\ to the list of UPROGS

Wait System Call:

1. No Children
    a. The ptable lock is acquired.
    b. The variable haveKids is set to 0.
    c. The process table is scanned, looking for a child that has the current process as a parent.
    d. Since the current process has no children, no child is found, and haveKids stays 0.
    e. Since haveKids is 0, the lock is released, and -1 is returned.
2. With Child Process
    a. The ptable lock is acquired.
    b. The variables haveKids is set to 0.
    c. The process table is scanned, looking for a child that has the current process as a parent.
    d. When it finds one, the child will either be in the ZOMBIE state or another process state.
        i. If it's not in the ZOMBIE state, the loop will keep repeating until one of the children is in this state.
    e. When a child is in the ZOMBIE state, the child's member variables are reset, state is set to UNUSED, the passed in status variable is set to the child's exit status, and resources are freed.
    f. The child's pid is returned.