The Study of Processes in UNIX
Lab. #2

Arif A. Balik

Undergraduate Student
Sytstems Programming
Department of Computer Science
Arel University
Büyükçekmece, İstanbul 34537
Email: arifbalik@outlook.com

March 11, 2019

Abstract

This report include analysis of some system calls in UNIX environment based on a program which provided in System Programming course. First section explains some system calls, second section answers some questions asked in the experiment paper, and third section suggests and implements some improvements and optimisations on the code.

This report is powered by LaTeX

Explanation of Some System and Library Functions

Difference between system and library functions

A system call is provided by the kernel and directly handled there, and a library functions are functions within programs, usually they are in the same level as shells.

write()

write is a system call used to output data to given channel (terminal, file etc.)

It has the prototype;

```
ssize_t write(int fildes, const void *buf, size_t nbytes);
```

where *fildes* is the channel that is where to write, *buf* is which to write and *nbytes* is how many bytes to write.

fork()

The system call *fork* creates an exact copy of current program. Takes no parameters and returns *0* for *child* process and child's id for *parent* process.

getpid()

The system call *getpid* returns the process id of currently running process.

getppid()

The system call *getppid* returns the process id of currently running process's parent process.

execl()

The system call *excl* replaces the current program with given program. It has the form;

```
int execl(const char *path, const char *arg, ...);
```

where *path* is the path for new executable file and *arg* is arguments to pass to the new process. Function only returns *-1* if there is an error. By convention the first argument to pass into new program should be the name of the program and list of arguments should end with *NULL*.

sleep()

In the labsheet, instructor specified the function *sleep* as a library function, but it is a system function in UNIX.[i]

It delays the system by given amount in seconds and returns *0* if the amount of time elapsed otherwise (in case of interruption) returns the amount of time left.

Theory of Operation

...

Q&A

Q1. *What is the purpose of this laboratory work?*

Q2. *How mony child processes created by the parent process in* ***mainprog.c****?*

Q3. *In* ***mainprog.c****, a few statements are executed by each created child process. Show these statements.*

[i]http://www.cs.miami.edu/home/geoff/Courses/CSC521-04F/Content/UNIXProgramming/UNIXSystemCalls.shtml

Q4. *In **mainprog.c**, to what program is child process switched? Show and explain the corresponding statement. Does a child process return to the **mainprog** after finishing another program (explain, why yes/no)?*

Q5. *Which statements are NOT executed by the parent process in **mainprog.c**?*

Q6. *What is the purpose of system call **wait()** in **mainprog.c**? What results are extracted by the parent process from this system call?*

Q7. *Is it possible for a child process to continue its work after the parent process terminates? What will the parent be for such a child? Prove your answer based on the results of the Part 2 of this lab work.*

*Q8. What is the purpose of directive **#define** in the program **procmemory.c**?*

Q9. *What is the meaning of the variables **etext**, **edata** and **end** in **procmemory.c**? Why are these variables declared with the word **extern**?*

Q10. *Suppose that a variable i was declared and assigned some value in the parent process before the creation of a child process. Will this variable be accessible to the child process? Will the parent process see the change made in this variable by the child process?*

Q11. *Is **malloc()** a system call or a library function? Can you guess it looking at your program?*

Q12. *What is the meaning of two parameters of the function **main()** in the program **child.c**? What is **argv[1]** in this program?*

Q13. *What is the purpose of system calls **getpid** and **getppid**?*

Q14. *Is it possible for a process to use more than one program?*

Q15. *Is it possible for two or more processes to use the same program?*

Q16. *What is the purpose of the system calls of the* ***exec*** *family? Dos this system call return any result (in case it succeeds or fails)?*

Optimization

Appendix