CS 130 Project 2: User Programs Design Document

Tianyi Zhang
School of Information Science and Technology
2018533074
zhangty2@shanghaitech.edu.cn

Haoran Dang
School of Information Science and Technology
2018533259
danghr@shanghaitech.edu.cn

0 PRELIMINARIES

0.1 Preliminary Comments

No preliminary comment for this project.

0.2 References

-

1 ARGUMENT PASSING

1.1 Data Structures

1.1.1 Copy here the declaration of each new or changed `struct` or `struct` member, global or static variable, `typedef`, or enumeration. Identify the purpose of each in 25 words or less.

- In file

1.2 Algorithms

1.2.1 Briefly describe how you implemented argument parsing. How do you arrange for the elements of `argv[]` to be in the right order? How do you avoid overflowing the stack page?

1.3 Rationale

1.3.1 Why does Pintos implement `strtok_r()` but not `strtok()`?

1.3.2 In Pintos, the kernel separates commands into a executable name and arguments. In Unix-like systems, the shell does this separation. Identify at least two advantages of the Unix approach.

2 SYSTEM CALLS

2.1 Data Structures

2.1.1 Copy here the declaration of each new or changed `struct` or `struct` member, global or static variable, `typedef`, or enumeration. Identify the purpose of each in 25 words or less.

- In file

2.1.2 Describe how file descriptors are associated with open files. Are file descriptors unique within the entire OS or just within a single process?

2.2 Algorithms

2.2.1 Describe your code for reading and writing user data from the kernel.

2.2.2 Suppose a system call causes a full page (4,096 bytes) of data to be copied from user space into the kernel. What is the least and the greatest possible number of inspections of the page table (e.g. calls to `pagedir_get_page()`) that might result? What about for a system call that only copies 2 bytes of data? Is there room for improvement in these numbers, and how much?

2.2.3 Briefly describe your implementation of the "`wait`" system call and how it interacts with process termination.

2.2.4 Any access to user program memory at a user-specified address can fail due to a bad pointer value. Such accesses must cause the process to be terminated. System calls are fraught with such accesses, e.g. a "`write`" system call requires reading the system call number from the user stack, then each of the call's three arguments, then an arbitrary amount of user memory, and any of these can fail at any point. This poses a design and error-handling problem: how do you best avoid obscuring the primary function of code in a morass of error-handling? Furthermore, when an error is detected, how do you ensure that all temporarily allocated resources (locks, buffers, etc.) are freed? In a few paragraphs, describe the strategy or strategies you adopted for managing these issues. Give an example.

2.3 Synchronization

2.3.1 The "`exec`" system call returns `-1` if loading the new executable fails, so it cannot return before the new executable has completed loading. How does your code ensure this? How is the load success/failure status passed back to the thread that calls "`exec`"?

2.3.2 Consider parent process P with child process C. How do you ensure proper synchronization and avoid race conditions when P calls `wait(C)` before C exits? After C exits? How do you ensure that all resources are freed in each case? How about when P terminates without waiting, before C exits? After C exits? Are there any special cases?

2.4 Rationale

2.4.1 Why did you choose to implement access to user memory from the kernel in the way that you did?

2.4.2 What advantages or disadvantages can you see to your design for file descriptors?

2.4.3 The default `tid_t` to `pid_t` mapping is the identity mapping. If you changed it, what advantages are there to your approach?

3 SURVEY QUESTIONS

In your opinion, was this assignment, or any one of the two problems in it, too easy or too hard? Did it take too long or too little time?

Did you find that working on a particular part of the assignment gave you greater insight into some aspect of OS design?

Is there some particular fact or hint we should give students in future quarters to help them solve the problems? Conversely, did you find any of our guidance to be misleading?

Do you have any suggestions for the TAs to more effectively assist students, either for future quarters or the remaining projects? Not yet.

Any other comments? Not yet.

CONTRIBUTORS

Task		Tianyi Zhang	Haoran Dang
Task 1 Argument Passing	Concept		
	Implementation		
	Debugging		
	Design Document		
Task 2 System Calls	Concept		
	Implementation		
	Debugging		
	Design Document		