Project 5: Multi-Thread in pi-OS

ENEE 447: Operating Systems — Spring 2017
Assigned Date: Tuesday, April 18th.
Due Date: **Friday, April 28th. @11:59 p.m.**

1.Abstract

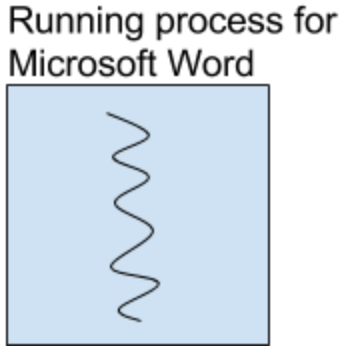
In project 4, we learned about **File Allocation Table** (**FAT**) file system. On top of the provided FAT file system interface, we implemented **virtual file system (VFS)**, which allow client applications to access files in different types of concrete file systems in a uniform way.

In project 5, we will learn about process and thread, and add system calls to make pi-OS to be multi-thread supported.

2.Introduction

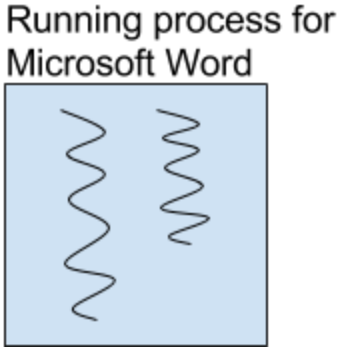
Process

A process is an executing (running) **instance** of an application or a program. What does it mean by an executing instance? What about an application? Let's walk through this with the Microsoft Word as an example. When you opened the Microsoft Word application, a process will be created for the program. The process is responsible for all the major editing feature provided by the Microsoft Word.



Thread

What about thread? A thread is a **path** of execution within a process. In the graph above, a running “MS word” process has single path of execution, which also known as the main thread of the “MS word” program. Multiple threads could be running in parallel within a process.



Back to the MS Word example, now we see two threads are running simultaneously within in a process. What is this telling us? In this case, the longer thread represents the main thread for the Microsoft Word process, while the short one, as a background thread represents the grammar correction feature in Microsoft Word. The ability of running multiple threads in a single process allow program to monitor user’s input and executing the grammar check at the same time.

Process vs Thread (textbook definition)

As we learned from the lecture, thread within the same process share the same address space. Address space is set of memory addresses that the process/thread can use without error. In other words, thread within the same process can read or write the same data structure and variables. Threads are able to communicate and exchange information via shared memory, because they are sharing the same address spaces. However, the communication between processes requires implementation of interprocess communication (IPC), which is outside the scope of this project.

Process vs Thread (practical world)

Even the textbook definition of process and thread set a clear line between them, however, in the industrial world, the textbook difference between them is less

significant. As we just discussed, the major difference between process and thread lies in whether they are sharing the same address space. In other word, whether they could communicate/sharing data with each other. For thread, it is very clear that **inter-thread communication** is through shared memory(because they are sharing the same address space). What about process? Processes between each other do not share the same address space, therefore, no way for them to communicate. Wrong, processes, indeed are able to communicate to one another via **inter-process communication** (IPC). With IPC supported, there is actually not much difference between process and thread with respect to functionality. Because both process and thread are trying to help program to run faster.

Process vs Thread (pi-OS)

What about pi-OS? In pi-OS, is there any difference between process and thread? Think about what is the textbook definition in difference between them?

In pi-OS, a process is exactly the same as a thread. Why?

The reason behind this lie into a concept related to address space, virtual memory. Virtual memory gives running process the abstraction that it has the entire address space (the entire memory space). Virtual memory also provides protection between processes, which mean one process could not read/write memory which is not belong to itself. Since pi-OS has not supported virtual memory yet, there is actually no notion of sharing address space. According to the textbook definition of the difference between thread and process, we can interpret process exactly as thread in the pi-OS.

3. Your mission

To add multi-thread, or multi-process capability into pi-OS, there are following steps:

- new system calls supported: fork() and wait()
- manage process states and relationships

The initial project defines the behavior of file operations in task 4, task 5 and task6, your implementation should behave like example output provided.

4. Implementation guide:

Please refer to ***Processes and Multi-thread in Pi-OS.pdf*** for implementation details.

5. Submission

No extension, any late submission (after **Friday, April 28th. @11:59 p.m.**) will result in penalty. If you submit one day late, 10% will be deducted from your project score. If you submit two days late, 20% will be deducted, etc.

What is inside your zip submission:

- Design/Documentation Report in **PDF format**. (include **screenshot** for working system call implementation with task 4, task 5 & task 6)
- All Project files, make sure it is compliable, any compile error automatically 10% off.

Demo:

- Make sure your code is working before submission, TA will download your code and compile there, if your code did **not compile**, and result of **demo not matching** our output. at least **10% project will be deducted**.
- Any reschedule for demo should be completed within the **same week** of demo.
- You need to answer questions during the demonstration. You can find example questions for project 4 on Piazza before demonstration. Failure of the demonstration would get you 15% deduction of score, and you would need to show again...

6. Honor Code

Copying code and idea is considering cheating and will be reported to Student Honor Council. The University of Maryland, College Park has a nationally recognized Code of Academic Integrity. This Code sets standards for academic integrity at Maryland for all undergraduate and graduate students. As a student you are responsible for upholding these standards for this course. It is very important for you to be aware of the consequences of cheating, fabrication, facilitation, and plagiarism. For more information on the Code of Academic Integrity or the Student Honor Council, please visit http://www.shc.umd.edu

7. Reference:

Context Switch Definition
http://www.linfo.org/context_switch.html
Advantages and Disadvantages of Threads
http://www.cs.iit.edu/~cs561/cs450/ChilkuriDineshThreads/dinesh%27s%20files/Advantages%20and%20disadvantages.htm
Processes vs Threads
http://www.programmerinterview.com/index.php/operating-systems/thread-vs-process/