Amirkabir University of Technology

Department of Computer engineering and information technology

Operating systems

Project 1

Author:

Kasra Mojallal

Professor:

Dr. Nastooh Taheri Javan

Brief explanation of the project:

- The aim of this project is to achieve a superficial view of the xv6 operating system and understand how the operating systems functions are designed and work. You will learn how to add system calls and how to test them. Last but not least, you will change the scheduling algorithm of the xv6 operating system and add a system call in order to change it in time of need.
- In order to implement the given tasks in xv6 first you should clone xv6 repository on your computer, delete the .git file and initialize a new one for your project. Please be aware that you should commit the changes you make to the project.

1. Questions about xv6

Please answer the following questions in detail. (Be informed that answering the following questions would help you understand xv6 better and you can do the other parts of the project easier.)

1. What is the default scheduling algorithm in xv6 and how does it work? (Look for it in proc.c)
2. What is the purpose of (allocproc – wait - fork) functions and how do they work?
3. What are the default attributes of proc? (you may need to add some in other parts of the project)
4. What does syscall function in syscall.c do?

2. Working with system calls

In this part you should add the following system calls to xv6.

For additional information visit here.

2.1 getChildren

in order to change the scheduling algorithm of xv6 (the main purpose of this project), first you have to understand how to add a system call and know how to test it. In this system call you should be able to get a list of the children (pid of the children) of the running process. This system call returns a string like “603”, which shows that the current process has two children with the pid of 6 and 3.

2.2 getChildren Test

After creating the system call, you should be able to test it. You have to create a getChildrenTest.c file. In this file you have to fork for several times and use this system call to see if the result is correct and this system call shows the correct number of the children for the running process.

For additional information about test files visit here.

3. Scheduling Algorithms

In this part, you have to change the basic scheduling algorithm of xv6.

3.1 changePolicy

You are going to add a new scheduling algorithm, so first you have to add a new system call in order to change between the default algorithm and the one you created. The name of this system call should be changePolicy.

3.2 Quantum

At each tick of the system, the scheduler gives the CPU to a process and at the end of that tick the process should release the CPU, but here we want to change that. You should define a variable called Quantum and give value to it. Then you should change a function that allows you to use the amount of quantum for context switching instead of doing it for every tick. (Due to the understanding you gained from xv6 questions in part 1, you should be able to find which function to change.)

3.3 Implementing Priority Algorithm

In this part you have to change the default algorithm so that you can set priority for processes and run the process that has the highest priority. You have to add a priority variable to your process and also you need another attribute called changeable priority, so that each time that the process runs, the value of the priority would be added to changeable priority, and the process that has the lowest amount of changeable priority would run.

Extra implementation

Before proceeding to the next part, you have to add some time variables to your process, as to measure time. We need creation time, running time, sleep time, waiting time and termination time. You have to add these attributes to your process in param.h. Now by the knowledge you gained from the previous parts you should be able to figure out how to add a new function called waitForChiled, that this function updates the time values of the process.

3.4 Implementing Multi level queue

After implementing other algorithms, you have to do this part. You have 3 different queues that each of them uses a different method. The first queue uses priority algorithm, the second one uses the default algorithm and the last one uses the same algorithm but with the given quantum. As illustrated in the class, you should be able to implement this part.

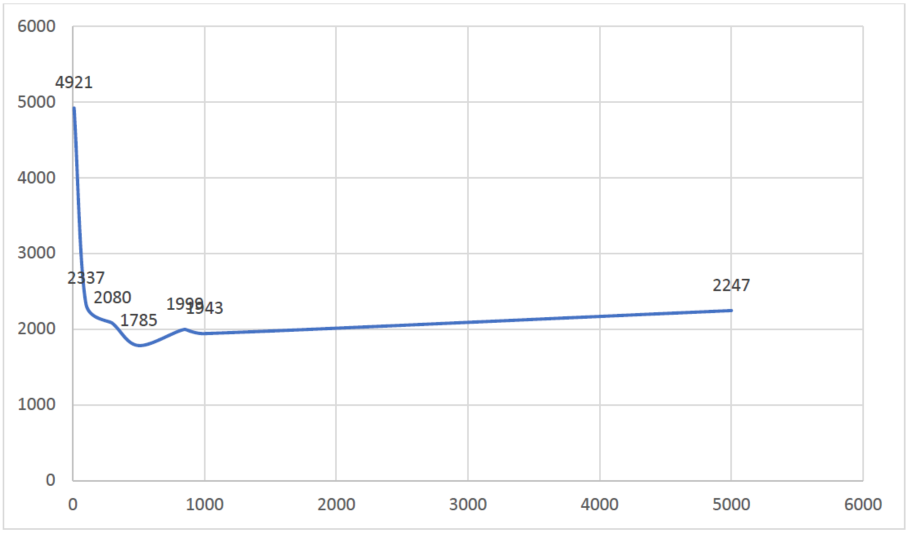
3.5 Test

Last but not least, you have to make some test files in order to check the performance of the system.

3.5.1

The first part is to create a test file for the default scheduling algorithm. The point is to change the amount of quantum and check the performance of the system with different quanta. It is highly recommended if you draw some charts for this part. For instance, take a look at the average waiting time of processes with different quanta.

Waiting Time:



3.5.2

In this part you just have to fork several times and run it with the priority algorithm policy, in order to check the result, you have to set a priority for your processes and check if the ones with the higher priority were finished earlier. This is done by measuring the time variables in proc and you should also create a system call for setting priority.

3.5.3

For the last sanity test, you have to use the multi-level queue algorithm. You are free to implement your test as you wish.

Please contact me if you have any questions.

Telegram: Kasramojallal

Email: Kasramojallal1@gmail.com