Lab2Answers

Name : Dhruv Subramanian

Question 3:
Hijack a process through stack smashing

a)Output with myvictim code:

Hello World!

I'm the first XINU app and running function main() in system/main.c.

I was created by nulluser() in system/initialize.c using create().

My creator will turn itself into the do-nothing null process.

I will create a second XINU app that runs shell() in shell/shell.c as an example.

You can do something else, or do nothing; it's completely up to you.

...creating a shell
My name's Dhruv Subramanian
Purdue Id : 0026458203

----------first function returns---------- 40

Process ID is:3,Myvictimglobal is:0

b) Output with stacksmash:

Hello World!

I'm the first XINU app and running function main() in system/main.c.

I was created by nulluser() in system/initialize.c using create().

My creator will turn itself into the do-nothing null process.

I will create a second XINU app that runs shell() in shell/shell.c as an example.

You can do something else, or do nothing; it's completely up to you.

...creating a shell
My name's Dhruv Subramanian
Purdue Id : 0026458203

----------first function returns---------- 40

Process ID is:3,Myvictimglobal 1 :Xinu trap!
exception 13 (general protection violation) currpid 3 (myvictim)
error code 00000000 (0)
CS FDE0008 eip 102353
eflags 10206
register dump:
eax 00000000 (0)
ecx 0000000A (10)
edx 00000000 (0)
ebx 00000000 (0)
esp 0FDEFFE8 (266272744)
ebp 0FDEFFE8 (266272744)
esi 00000000 (0)
edi 00000000 (0)

panic: Trap processing complete...

Procedure:

Attackermalwares' PID is the same as that of myvictim, which is 15. This is because myvictim or one of its nested functions points to the myattackermalware, and on getting myvictimglobal = 1, we haven't actually called myattackermalware. Only myvictim, and its nested functions have been called, implying that the process ID is that of myvictim.

The reason behind the xinu trap is that myattackermalware doesn't have a return address on the runtime stack as it hasn't been called directly. It is accessed when the return address of one of the nested functions within myvictim points to it.

When myattackermalware finishes executing, a xinu trap occurs, because myattackermalware has a garbage return address, or a corrupted one which causes xinu to crash.

Question 4:

Implementation: I followed the procedure given in the lab handout and in resched.c I initialized a global variable which would store the cpu time after each successive cycle. I stored the difference between the clktimefine and store in order to keep updating the cpuused. This was followed by context switching in and out. Then I reran the testcases in 4.1 and 4.3 of the previous lab.

Output for Lab answers for 4.1 and 4.3:

Hello World!

I'm the first XINU app and running function main() in system/main.c.

I was created by nulluser() in system/initialize.c using create().

My creator will turn itself into the do-nothing null process.

I will create a second XINU app that runs shell() in shell/shell.c as an example.

You can do something else, or do nothing; it's completely up to you.

...creating a shell
My name's Dhruv Subramanian
Purdue Id : 0026458203
----------Question 4,Lab 2----------
----------Running 4.1 and 4.3 for 4 of Lab 2----------
AAAAAP
BBBBBP
CCCCCP
DDDDDTime is 51
Time is 51
Time is 51
Time is 51
AAAAAP
BBBBBP
CCCCCP
DDDDDTime is 51
Time is 51
Time is 51
Time is 51

These results are calculated in main,and the time taken during these processes is printed out as seen above in the output. As expected the, all the times even for processes with different priorities are the same. This is because, the time taken in terms of cpuused is taken into account and not the priority itself.

Question 5: Fair time Scheduling

Testscases:

1. Output:(CPUintensive)

Hello World!

I'm the first XINU app and running function main() in system/main.c.

I was created by nulluser() in system/initialize.c using create().

My creator will turn itself into the do-nothing null process.

I will create a second XINU app that runs shell() in shell/shell.c as an example.

You can do something else, or do nothing; it's completely up to you.

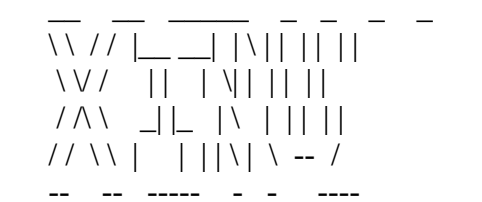
...creating a shell
My name's Dhruv Subramanian
Purdue Id : 0026458203
----------Question 4,Lab 2----------
----------Running 4.1 and 4.3 for 4 of Lab 2----------

----------Question 5 (Part 1), Lab 2----------

----------Question 5 (Part 2), Lab 2----------

----------Question 5 (Part 3), Lab 2----------

PID:3 ,i:0 ,priorityPID:4 ,i:0 ,priority is:PID:5 ,i:0 ,priority is:PID:6 ,i:0 ,priority is: is:1 ,time slice:10
1 ,time slice:10
1 ,time slice:10
1 ,time slice:10
PID:PID:4 ,iPID:5 ,iPID:6 ,i3 ,i:1 ,priority is:1 ,time slice:10
:1 ,priority is:1 ,time slice:10
:1 ,priority is:1 ,time slice:10
:1 ,priority is:1 ,time slice:10
PID:3 ,i:2 ,priority is:1 ,time slice:10
PID:4 ,i:2 ,priority is:1 ,time slice:10
PID:5 ,i:2 ,priority is:1 ,time slice:10
PID:6 ,i:2 ,priority is:1 ,time slice:10
PID:3 ,i:3 ,priority is:1 ,time slice:10
PID:4 ,i:3 ,priority is:1 ,time slice:10
PID:5 ,i:3 ,priority is:1 ,time slice:10
PID:6 ,i:3 ,priority is:1 ,time slice:10



Welcome to Xinu!

xsh $ PID:4 ,i:4 ,priority is:1 ,time sliPID:5 ,i:4 ,priority is:1 ,time slicPID:6 ,i:4 ,priority is:1 ,time

slice:10
cCPU time:756
e:10
CPU time:756
e:10
CPU time:756
PID:3 ,i:4 ,priority is:1 ,time slice:10
CPU time:757

2. Output: (IOintensive)

Hello World!

I'm the first XINU app and running function main() in system/main.c.

I was created by nulluser() in system/initialize.c using create().

My creator will turn itself into the do-nothing null process.

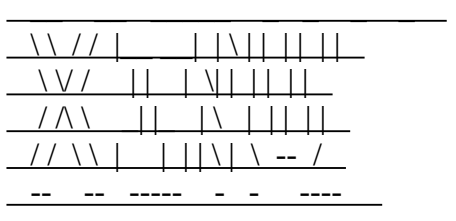
I will create a second XINU app that runs shell() in shell/shell.c as an example.

You can do something else, or do nothing; it's completely up to you.

...creating a shell
My name's Dhruv Subramanian
Purdue Id : 0026458203

----------Question 5 (Part 2), Lab 2----------

PID:3, i:0, The process' priority:1, slice:10
PID:4, i:0, The process' priority:1, slice:10
PID:5, i:0, The process' priority:1, slice:10



Welcome to Xinu!

xsh $ PID:6, i:0, The process' priority:1, slice:10

PID:3, i:1, The process' priority:1, slice:10
PID:4, i:1, The process' priority:1, slice:10
PID:5, i:1, The process' priority:1, slice:10
PID:6, i:1, The process' priority:1, slice:10
PID:3, i:2, The process' priority:1, slice:10
PID:4, i:2, The process' priority:1, slice:10
PID:5, i:2, The process' priority:1, slice:10
PID:6, i:2, The process' priority:1, slice:10
PID:3, i:3, The process' priority:1, slice:10
PID:4, i:3, The process' priority:1, slice:10
PID:5, i:3, The process' priority:1, slice:10
PID:6, i:3, The process' priority:1, slice:10
PID:3, i:4, The process' priority:1, slice:10
The CPU time consumed is : 13
PID:4, i:4, The process' priority:1, slice:10
The CPU time consumed is : 13
PID:5, i:4, The process' priority:1, slice:10
The CPU time consumed is : 13
PID:6, i:4, The process' priority:1, slice:10
The CPU time consumed is : 13

3. Output:(Half and Half)

Hello World!

I'm the first XINU app and running function main() in system/main.c.

I was created by nulluser() in system/initialize.c using create().

My creator will turn itself into the do-nothing null process.

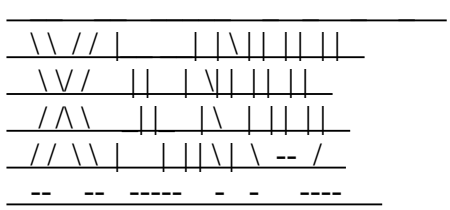
I will create a second XINU app that runs shell() in shell/shell.c as an example.

You can do something else, or do nothing; it's completely up to you.

...creating a shell
My name's Dhruv Subramanian
Purdue Id : 0026458203

----------Question 5 (Part 3), Lab 2----------
PID:11 ,i:0 ,priority is:1 ,time slice:10
PID:11 ,i:1 ,priority is:1 ,time slice:10
PID:11 ,i:2 ,priority is:1 ,time slice:10
PID:11 ,i:3 ,priority is:1 ,time slice:10
PID:11 ,i:4 ,priority is:1 ,time slice:10
CPU time:721
PID:12 ,i:0 ,priority is:1 ,time slice:10
PID:12 ,i:1 ,priority is:1 ,time slice:10

PID:12 ,i:2 ,priority is:1 ,time slice:10
PID:12 ,i:3 ,priority is:1 ,time slice:10
PID:12 ,i:4 ,priority is:1 ,time slice:10
CPU time:721



Implementation:
Points to note:
ready state queue: refers to the list of ready processes stored as a queue in ascending order of priorities

Once the current process is ready to be scheduled, resched is called.

The basic idea is to have the process table maintained by the operating system to facilitate context switching and scheduling. The process table is identified by its unique process ID. A check is performed in the scheduler to see if the process is in the current or ready state. If it is in the current state, a comparison is done to check if the current process' cpu time is less than that of the the first process in the ready list. If it is, the current process continues, else, the the process in the ready list is dequeued and set as the current process.

The time of a cycle is then calculated and updated.
Context switch is called which changes the state of process.

In the actual implementation, I passed cpuused as the argument to the insert function instead of the priority variable which would be a means to reverse the order of the queue.

The way resched works is that there is already a null process created by xinu which initializes the system. It has a a pid of zero and priority zero.

Bonus Question:

A possible solution that may mitigate the problem of fair scheduling is that instead of using the total clock time of cpu used, we can compute the time slice of a particular cycle. This would be smaller than the whole cycle and so it would generate more of a possibility when it comes to fair scheduling. This

would give all processes a better chance rather than giving new processes elevated priority over longer running processes, hence reducing the possibility of starving them.