Akash Lankala
CS 354
Project 2
Thursday, February 20th, 2020

Dynamical behavior of XINU's process scheduler

3.2 Experiments

Find where the time budget `QUANTUM` is defined. Note its value. (put in a comment the previous value out so that you keep a history).

> `QUANTUM` is defined in `kernel.h` with the defined value as 2.

(a) After changes to main(), what output to CONSOLE do you observe? Check what the constant INITPRIO is which determines the priority of the process executing main().

> A's and B's were printed repeatedly and in equal number because both of them are of same priority 10 and each process runs for 30 msec. About 345 A's were printed followed by 345 B's, and so on.
> INITPRIO is defined as constant 20 in `process.h`. This is the initial process priority.

(b) Rerun experiment (a) with QUANTUM set to 5 msec. Discuss your finding.

> A's and B's ran a shorter time compared to (a) for 5 msec. Only about 58 A's were printed followed by 58 B's.

(c) Go back and change the priority of the process printing 'B' to `15`. Discuss your finding.

> It is only printing B's to console. This is expected because the process printing B has a higher priority (15) than the process printing A (10) and timeslicing (QUANTUM) does not apply here.

(d) Go back and change the priority of the process printing 'A' to `25`. Discuss your finding.

Now only A's are getting outputted to console. This is due to the process printing A now having a higher priority (25) than the one printing B (15).

(e) Go back to the set-up of experiment (a) and change the priority of the process printing 'A' to `20`. Discuss your finding. Explain the difference between (d) and (e).

With the priority of process printing A set to 20 and the priority of process B set to 15, only A's are printed to the console. This is because the priority of the process printing A is higher than that printing B. Compared to (d), the priority of the process printing A has been lowered to 20 from 25. But this still has a higher priority than the process printing B (15).

(f) Modify XINU's `chprio()` so that it returns `SYSERR` when 0 or a negative priority value is passed as an argument. Explain why this modification is necessary to preserve an important property of the `NULL` process (or `IDLE` process in Linux/UNIX and Windows) in today's operating systems.

Check `chprio()` for modifications.

This modification is necessary so that the priority of this process does not conflict with the null process.

4. Kernel mod: adding new system calls

(a) Create a new system call, `pid32 getppid(pid32),` that allows a process to query the PID of the parent of the process given by the argument. Follow sanity check of `chprio()` to verify that the PID passed in the argument is valid. If not, return `SYSERR`. Place code of `getppid()` in `getppid.c` under `system/`. Check that your system call works correctly.

Check `getppid.c` for the code.

Akash Lankala
CS 354
Project 2
Thursday, February 20th, 2020

(b) Create a new system call, `uint32 gettmslice(void)`, that allows a process to check how much time (in msec) remains in its time slice. Put code of `gettmslice()` in `gettmslice.c` under `system/`. Check that your system call works correctly.

Check `gettmslice.c` for the code.

5. Trapped XINU system calls

5.1, 5.2, 5.3

Check files `wgetpid.c`, `wgetprio.c`, function `_Xint33` in `intr.S`, as well as `main.c` for these additions to the code.

Bonus problem

Check `wgetppid.c` for this addition.