Lab 4: Set Priority and List Process Info

Instructor: Yaoqing Liu
TA: Garegin Grigoryan

Download Instructions and xv6

- Go to moodle to download this instruction file
- Login to odin

$ssh YourName@odin.cslabs.clarkson.edu

$cd ~/cs444-s18/Lab4-5

- Download xv6.tar.gz file to your work directory Lab4-5

$wget http://people.clarkson.edu/~liu/CS444/Spring18/xv6.tar.gz

- Unzip it

$tar –xzvf xv6.tar.gz

Your tasks

1. Add “priority” field to the processes in xv6

   - Initialize priorities with 0

2. Implement ps_yourname()

   - Shows the list of current processes, their process IDs and priorities

3. setpriority_yourname(pid, pr)

   - Sets the priority of the process to priority ***pr*** *for a specified the process ID* ***pid***

4. Test ps_yourname() and setpriority_yourname(pid, pr) with a user program

1. Adding priority to a process

- Add ***priority*** to ***proc*** structure in proc.h
- Initialize priority with 0 in the process allocation function (***allocproc()*** in **proc.c**)
- Modify ***fork()*** in **proc.c** correctly (priority should be copied to the newly cloned process)

2. Implementing ps_yourname() system call

- ps_yourname() shall print the process list:

```
$ ps_grigorg
pid    name    state   priority
1       init    2       0
2       sh      2       0
3       ps_grigorg      4       0
```

- Implement ps() in proc.c file and edit other system files (**see the next slide**)
- Hint: in proc.c, use **cprintf(...)** function to print process information for every process in **ptable**
- For iterating through **ptable,** use a for loop like in the **scheduler()** system call.
- Don't print lines when pid = 0

Help: adding a new system call sc

1. **userspace/usys.S**
   - SYSCALL(*sc*)
2. **syscall.h**
   - #define SYS_*sc* <newID>
3. **sysproc.c**
   - int sys_*sc*(void) { return *sc*(); }
4. **userspace/user.h**
   - int *sc*(void);
5. **syscall.c**
   - extern int sys_sc(void);
   - [SYS_*sc*] sys_*sc*,
6. **defs.h**
   - int *sc*(void);
7. **Implement** int *sc*(void) {….your code...} in **proc.c**

3. Implement setpriority_yourname(pid, pr)

- User should be able to change the priority of a process with process ID ***pid*** to priority ***pr***
- 2 arguments (***pid*** and ***pr***) must be passed using the command line
- If process ID does not exists, system call returns **-1**
- Implement ***setpriority_yourname(..)*** in proc.c
- Example of passing the arguments is in the next slide

Example: Passing Arguments

- System call **kill(int pid)**: make directory
- **userspace/kill.c**

```
int main(int argc, char *argv[]){
  .......
    kill(atoi(argv[i]));
  .......
}
```

- **sysproc.c**

```
Int sys_kill(void)
{
  int pid;
 if(argint(0, &pid) < 0)
    return -1;
  return kill(pid);
}
```

- **proc.c**

```
int kill(int pid){.....}
```

4. Test with User Programs

- Implement two user programs (in userspace/):
  - **ps_yourname.c** calls ps() system call
  - **setpriority_yourname.c** takes two arguments (process ID, priority) and changes the priority of the process
    - If set priority successfully, return the new priority
    - Otherwise, return -1 and prints an error message

Example output of user programs

- ```
  $ ps_grigorg
  pid    name    state   priority
  1       init    2       0
  2       sh      2       0
  3       ps_grigorg      4       0

  $ setpriority_grigorg 2 3
  Priority of the process with ID 2 is set to 3
  $ ps_grigorg
  pid    name    state   priority
  1       init    2       0
  2       sh      2       1
  6       ps_grigorg      4       1

  $ setpriority_grigorg 100 100
  Process with ID 100 does not exist
  ```
  ```

Submission

- Take screenshots for your modifications, compiling process and the running results
- Combine them into YourName.pdf and upload it to moodle
- Due: Feb 12 (Monday), 11:55pm

Prepare for Next Lab

- Once you're able to set priority and list existing process info, next week you will need to implement a priority-based scheduler
- Start early, it will be tricky
- Replace xv6's round-robin scheduler with the new scheduler
  - Valid priority range: 0-200, inclusive
  - The smaller value represents higher priority
  - The default priority: 0
  - The scheduler selects the process with the highest priority for execution
  - If multiple processes have the same priority, use round robin scheduling algorithm to execute them in turn

Thank You!