CS 1550

Week 6 – Project 2

Teaching Assistant

Xiaoyu (Veronica) Liang

CS 1550 – Project 2

- **Due**: Friday, October 18th @11:59pm
- **Late**: Sunday, October 20th@11:59pm with 10% reduction per late day

Create a new process – fork()

- int fork(void)
  - Create a new process
  - No argument
  - Returns 0 to the child process
  - Returns child’s pid to the parent process (>0)
  - After creation of the process, both parent and child processes start execution from the next instruction following the fork() system call.

Create a new process – fork()

```
int pid = fork();

if (pid < 0) {
   fprintf(stderr, "Fork failed\n");
   exit(1);
}

if (pid == 0) {
   printf("This is the child\n");
   exit(0);
}

if (pid > 0) {
   printf("This is parent. The
   child's pid is %d\n", pid);
   exit(0);
}
```

Create a new process – fork() Example

- See whiteboard, demo

Project 2 – constraints

- Safe apartment inspection problem
  - A tenant cannot view an apartment without an agent
  - An agent cannot open the apartment without a tenant
  - An agent leaves when no more tenants are in the apartment
  - An agent cannot leave until all tenants in the apartment leave
  - Once an agent opens the apartment, she can show the apartment to at most ten tenants, and
  - At most one agent can open the apartment at a time
- Always satisfies the above constraints
- Under no conditions will a deadlock occur

Project 2

- The tenant process
- The agent process
- Tenant arrival process
- Agent arrival process

- tenantArrives()
- viewApt()
- tenantLeaves()
- agentArrives()
- openApt()
- agentLeaves()

Project 2 – arguments

- Command-line arguments:
  - -m: number of tenants
  - -k: number of agents
  - -pt: probability of a tenant immediately following another tenant
  - -dt: delay in seconds when a tenant does not immediately follow another tenant
  - -st: random seed for the tenant arrival process
  - -pa: probability of an agent immediately following another agent
  - -da: delay in seconds when an agent does not immediately follow another agent
  - -sa: random seed for the agent arrival process

Project 2 – Testing your program

- You can find a modified kernel with semaphore implementation in the following files (also can be found in course web, TA's webpage):
  - /u/OSLab/original/bzImage
  - /u/OSLab/original/System.map
  - /u/OSLab/original/sem.h
  - /u/OSLab/original/unistd.h (to compile your programs against)
    - cp unistd.h to linux-2.6.23.1/include/asm/
- Compile aptsim.c (the same way how you compile trafficsim)
  - gcc –m32 –o aptsim –I /PATH/TO/linux-2.6.23.1/include aptsim.c
- Log in your QEMU virtual machine:
  - Copy the kernel image with cs1550 semaphore implementation (System.map and bzImage) to QEMU (/boot/bzImage-devel, /boot/System.map-devel)
  - lilo
  - Reboot, select devel mode from the boot loader menu.
- Copy aptsim to QEMU and run the program by
  - ./aptsim

Project 2 – Submission

- Well commented aptsim.c file
- A brief, intuitive explanation of why your solution is
  - fair (maximum 10 tenants per agent),
  - deadlock, and
  - starvation free.

CS 1550

Week 6 – Project 2

Teaching Assistant

Xiaoyu (Veronica) Liang