Brian Duenas

CSE 460

Dining Philosophers and XV6 Process Priority

1. Dining Philosophers and Deadlock

**Q:** Try dine1.cpp. Type ^C to check the number of philosophers eating. Run it for some time. What conclusion can you draw on the number of philosophers that can eat at one time?

**Output:**

****

**A:** Only one philosopher can eat at a time.

**Q:** Compile and run dine2.cpp, and repeat the experiment as above. What is the maximum number of philosophers who can eat simultaneously?

**Output:**

****

****

**A:** Only a maximum of two philosophers can eat simultaneously.

**Q:** Add a delay statement like SDL\_Delay ( rand() % 2000 ); right after the take\_chops( l ) statement in the philosoper() function. Run the program for a longer time. What do you observe?

**Output:**

****

****

**A:** The philosophers arrive at a deadlock because each has one of the chopsticks.

**Q:** Implement this mechanism as discussed in class and call your program dine3.cpp. Repeat the above experiment to see whether deadlock occurs and what the maximum number of philosophers can dine simultaneously.

**Code:**

****

****

****

**Output:**

****

**A:** The maximum number of philosophers that can eat at one time is two.

2. XV6 Process Priority

**Q:** Do the experiment as described. Summarize all the steps, including those not presented explicitly above.

**Process:**

1) Add priority to struct proc in proc.h

2) Assign default priority in allocproc() in proc.c

3) Modify cps() in proc.c so it prints out the process priority

4) Modify foo.c so that it loops for a much longer time before exit.

5) Add the function chpr() (meaning change priority) in proc.c

6) Add sys\_chpr() in sysproc.c

7) Add name to syscall.h

8) Add function prototype to defs.h

9) Add function prototype to user.h

10) Add function call to sysproc.c

11) Add call to usys.S

12) Add call to syscall.c

13) Create the user file nice.c with which calls chpr

**Output:**

****