Team 4

Anthony Redamonti

Adrian Bernat

Dana Dippery

Michael Rice

Ousmane Barry

Syracuse university

Lab 2

CIS-657 Principles of operating systems

prof. mohammed abdallah

2/2/2021

Part 1: The Queuetab

The following main.c program creates three processes, places them on the ready queue, and prints the contents of the queuetab.

/\* main.c - main\*/

#include <xinu.h>

void procA(void), procB(void), procC(void);

/\* main - Example of creating processes in Xinu \*/

void main(void)

{

resume(create(procA, 1024, 20, "p 1", 0));

resume(create(procB, 1024, 20, "p 2", 0));

resume(create(procC, 1024, 20, "p 3", 0));

int i = 0;

while(i < NQENT){

    kprintf("i: %d, Key: %d, Prev: %d, Next: %d\n", i, queuetab[i].qkey, queuetab[i].qprev, queuetab[i].qnext);

    i++;

    }

}

void procA(void)

{

    while(1);

}

void procB(void)

{

    while(1);

}

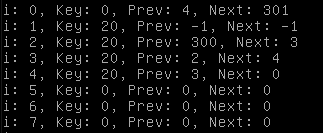
void procC(void)

{

    while(1);

}

The contents of each qentry are printed to the console shown below. The first column is the index into the queuetab array. The next column is the key followed by the previous and next pointers. The first element is the NULL process and has a key of zero. The key is zero because the priority of the NULL process is zero (lowest possible priority). The NULL process has its prev and next pointers assigned to 4 and 301 respectively.



The next entry in the array has a key value of 20 with prev and next pointers assigned to -1 (NULL) because this queue node represents the main process, which is the current process. Because it is the current process, it will no longer exist on the ready queue.

Indexes 2 through 4 refer to the processes created by the main function. The node order (by pointer) is:

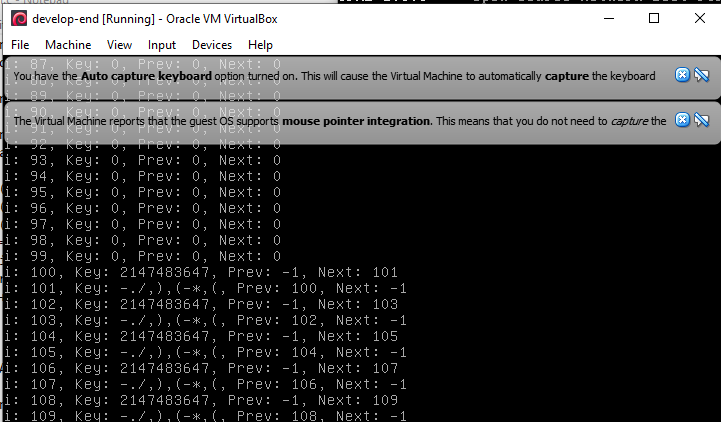
* 300 (head pointer)
* 2 (user process)
* 3 (user process)
* 4 (user process)
* 0 (null process)
* 301 (tail pointer)

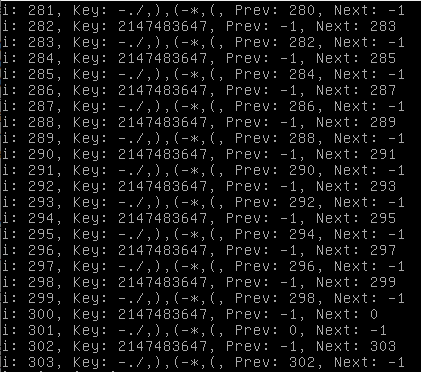
The rest of the processes from 0 to NPROC-1 (NPROC equals 100 as per conf.h) have key, prev and next pointers equal to 0 because no further processes have been created.

Then entries from 100 to 299 refer to semaphores. By default (in conf.h), 100 semaphores are allowed in the Xinu system and there are two entries per semaphore. Finally, entries 300 through 303 contain the head and tail pointers for the process tables.

|  |  |  |  |
| --- | --- | --- | --- |
| **Index** | **Key** | **Prev** | **Next** |
| 0 | 0 | 4 | 301 |
| 1 | 20 | -1 | -1 |
| 2 | 20 | 300 | 3 |
| 3 | 20 | 2 | 4 |
| 4 | 20 | 3 | 0 |
| 5 | 0 | 0 | 0 |
| … |  |  |  |
| 99 | 0 | 0 | 0 |
| … semaphores |  |  |  |
| 300 | 2147483647 | -1 | 2 |
| 301 | -2147483648 | 0 | -1 |
| 302 | 2147483647 | -1 | 303 |
| 303 | -2147483648 | 302 | -1 |

When the elements in the array are outside the range of NPROC-1, the qid portion of the queuetab has been entered. See the transition below.





The key in the head of the qid is the MAXKEY. The MAXKEY is defined as the largest value a signed 32-bit integer can product (2,147,483,647). The element following the head is the tail which has a key of MINKEY. The MINKEY should have a value of -2,147,483,648 but instead has a value of “-./,),(-\*,(,”. The prev of the head and next of the tail are -1 (NULL) as described in the textbook. The unused queues are empty; i.e. queuehead(q).next is pointing to queuetail(q), and queuetail(q).prev is pointing to queuehead(q). The second to last queue is the only queue being used as it contains one qentry located at index 0.

Part 2: Using a Semaphore

Below is the program written to create two processes “m1” and “m2”. Both processes share a semaphore “sem”. M1 will print out the first 21 numbers of integer ‘i’ then block on the semaphore. M2 will then print “signaler is running” on a new line. Then m2 will increment the semaphore 5 times, allowing m1 to print the next 5 numbers. To ensure that m1 runs first, it is given a priority of 40, and m2 is given a priority of 20. The pattern will continue until the m1 integer has incremented integer ‘i’ 2,000 times. Then m1 kills m2. Output of the code is on the next page.

/\* main.c - main \*/

#include <xinu.h>

void m1();void m2();

sid32 sem;

pid32 m1pid, m2pid;

void main(void)

{

    sem = semcreate(20);

    m1pid=create(m1, 1024, 40,"m1", 0);

    m2pid=create(m2, 1024, 20,"m2", 0);

    resume(m1pid);

    resume(m2pid);

    return OK;

}

void m1(){

    int32 i;

    for (i = 1; i <= 2000; i++) {

        kprintf("%d ", i);

        wait(sem);

    }

    kill(m2pid);

}

void m2(){

    while(1) {

        kprintf("signaler is running\n");

        signaln(sem,5);

    }

}

