Midterm Exam

CIS-657 PRINCIPLES OF OPERATING SYSTEMS

PROF. MOHAMMED ABDALLAH

1/28/2021

Anthony Redamonti
SYRACUSE UNIVERSITY

2.1: xsh_create.c

The following xsh_create.c program creates a shell command that creates processes. The argument to the command is the priority of the process and is optional. If no priority is provided, then the default value of the priority if 20. The runforever process is then created and resumed.

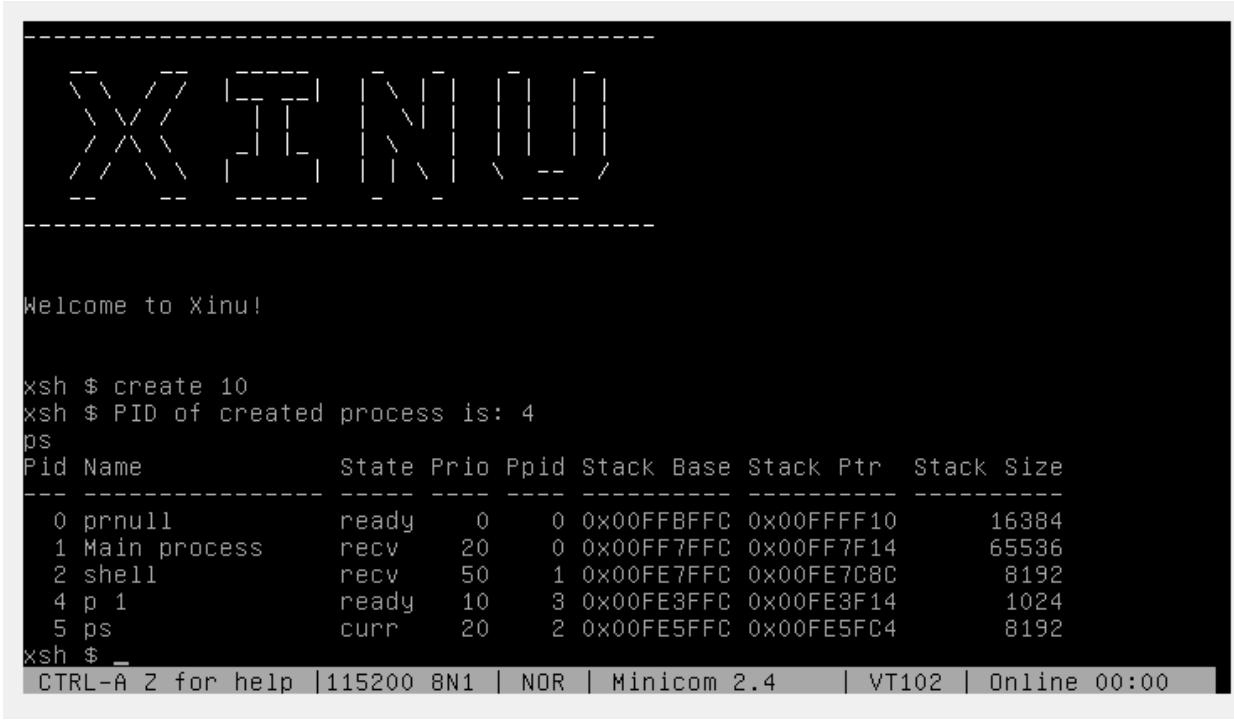
```
/* xsh_create.c - xsh_create */

#include <xinu.h>
#include <stdio.h>
#include <string.h>

/*------------------------------------------------------------------------
 * xsh_create - shell command to create a process
 *------------------------------------------------------------------------
 */
shellcmd xsh_create(int nargs, char *args[])
{
    pid32   pid;            /* return PID           */
    pri16   priority;       /* process priority > 0     */
    char    ch;             /* next character of argument   */
    char    *chptr;         /* walks along argument string  */

    /* Output info for '--help' argument */

    if (nargs == 2 && strncmp(args[1], "--help", 7) == 0) {
        printf("Usage: %s Priority\n\n", args[0]);
        printf("Description:\n");
        printf("\tcreates a process\n");
        printf("Options:\n");
        printf("\tPriority \tthe priority of a process to create\n");
        printf("\t\t If no priority provided, a default value\n");
        printf("\t\t of 20 is used\n");
        printf("\t--help\tdisplay this help and exit\n");
        return OK;
    }

    /* Check argument count */

    if (nargs > 2) {
        fprintf(stderr, "%s: incorrect argument\n", args[0]);
        fprintf(stderr, "Try '%s --help' for more information\n", args[0]);
        return SYSERR;
    }
```

```
/* compute priority from argument string */
    if (nargs == 2) {
        chptr = args[1];
        ch = *chptr++;
        priority = 0;
        while(ch != NULLCH) {
            if ( (ch < '0') || (ch > '9') ) {
                fprintf(stderr, "%s: non-digit in process ID\n",
                    args[0]);
                return 1;
            }
            priority = 10*priority + (ch - '0');
            ch = *chptr++;
        }

        pid = create(runforever, 1024, priority, "p 1", 0);
        if (pid == SYSERR) {
            fprintf(stderr, "%s: cannot create process %d\n",
                args[0], priority);
            return 1;
        }
        resume(pid);
    }

    /* No argument so use priority default value of 20 */

    if (nargs == 1){
        pid = create(runforever, 1024, 20, "p 1", 0);
        if (pid == SYSERR) {
            fprintf(stderr, "%s: cannot create process %d\n",
                args[0], priority);
            return 1;
        }
        resume(pid);
    }
    return 0;
}
```

The runforever process is defined in runforever.c shown below. It will print out the current PID and then call an infinite while-loop.

```c
/* runforever.c - runforever */

#include <xinu.h>

/*------------------------------------------------------------------------
 *  runforever  -  run forever in an infinite while-loop
 *------------------------------------------------------------------------
 */
void runforever(void)
{
    printf("PID of created process is: %d\n", currpid);
    while(1);
}
```

Here is the output in the terminal:



Part 2.2: xsh_createsleep.c

The following xsh_createsleep.c program creates a shell command that creates processes that will run forever after sleeping for 10 seconds. The argument to the command is the priority of the process and is optional. If no priority is provided, then the default value of the priority if 20. The runafterwait process is then created and resumed.

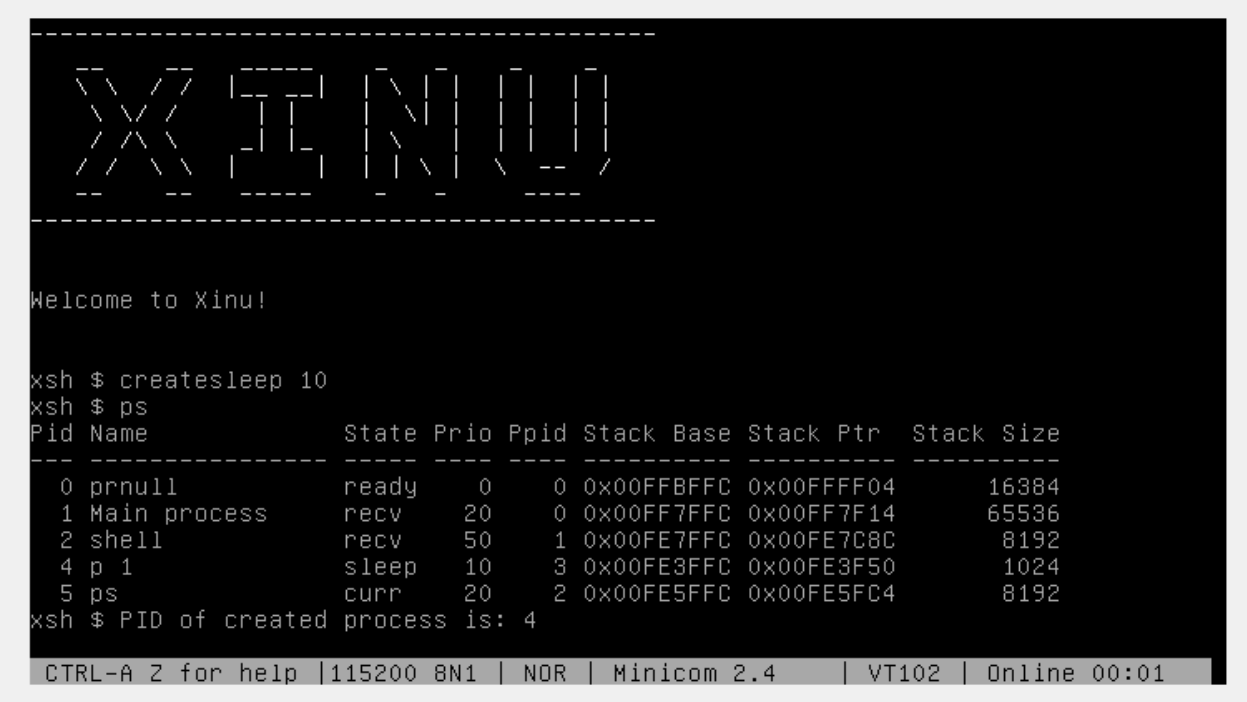
```
/* xsh_create.c - xsh_create */

#include <xinu.h>
#include <stdio.h>
#include <string.h>

/*------------------------------------------------------------------------
 * xsh_createsleep - shell command to create a process
 *------------------------------------------------------------------------
 */
shellcmd xsh_createsleep(int nargs, char *args[])
{
    pid32   pid;            /* return PID           */
    pri16   priority;       /* process priority > 0     */
    char    ch;             /* next character of argument   */
    char    *chptr;         /* walks along argument string  */

    /* Check argument count */

    if (nargs > 2) {
        fprintf(stderr, "%s: incorrect argument\n", args[0]);
        fprintf(stderr, "Try '%s --help' for more information\n", args[0]);
        return SYSERR;
    }


/* compute priority from argument string */
    if (nargs == 2) {
        chptr = args[1];
        ch = *chptr++;
        priority = 0;
        while(ch != NULLCH) {
            if ( (ch < '0') || (ch > '9') ) {
                fprintf(stderr, "%s: non-digit in process ID\n",
                    args[0]);
                return 1;
            }
            priority = 10*priority + (ch - '0');
```

```
            ch = *chptr++;
        }

        pid = create(runafterwait, 1024, priority, "p 1", 0);
        if (pid == SYSERR) {
            fprintf(stderr, "%s: cannot create process %d\n",
                args[0], priority);
            return 1;
        }
        resume(pid);
    }

    /* No argument so use priority default value of 20 */

    if (nargs == 1){
        pid = create(runafterwait, 1024, 20, "p 1", 0);
        if (pid == SYSERR) {
            fprintf(stderr, "%s: cannot create process %d\n",
                args[0], priority);
            return 1;
        }
        resume(pid);
    }
    return 0;
}
```

The runafterwait process is defined in runafterwait.c shown below. It will sleep for 10 seconds, print out the current PID and then call an infinite while-loop.

```
/* runforever.c - runforever */

#include <xinu.h>

/*------------------------------------------------------------------------
 *  runforever  -  run forever in an infinite while-loop
 *------------------------------------------------------------------------
 */
void runafterwait(void)
{
    sleep(10);
    printf("PID of created process is: %d\n", currpid);
    while(1);
}
```

Here is the output from the terminal:



Part 2.3: xsh_psready.c

The following xsh_psready.c program creates a shell command that only prints out the ready processes in the system. The shell command was a copy of xsh_ps.c but with only a few lines edited.

In xsh_psready.c in line 53:

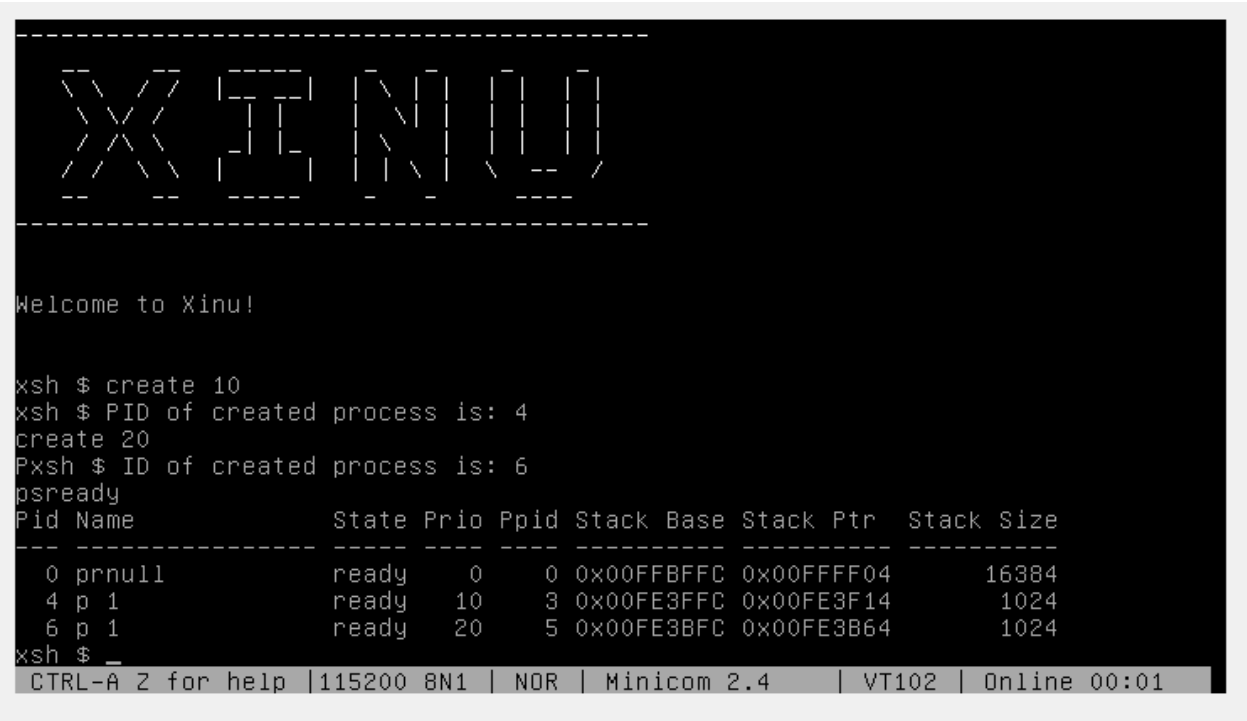
```
        if (prptr->prstate != PR_READY) {  /* skip slots that are not ready
processes  */
```

Here is the entire xsh_psready function:

```
/* xsh_psready.c - xsh_psready */

#include <xinu.h>
#include <stdio.h>
#include <string.h>

/*------------------------------------------------------------------------
 * xsh_psready - shell command to print the process table
 *------------------------------------------------------------------------
 */
shellcmd xsh_psready(int nargs, char *args[])
{
    struct  procent *prptr;     /* pointer to process       */
    int32   i;          /* index into proctabl      */
    char *pstate[]  = {     /* names for process states */
        "free ", "curr ", "ready", "recv ", "sleep", "susp ",
        "wait ", "rtime "};

    /* For argument '--help', emit help about the 'ps' command  */

    if (nargs == 2 && strncmp(args[1], "--help", 7) == 0) {
        printf("Use: %s\n\n", args[0]);
        printf("Description:\n");
        printf("\tDisplays information about running processes\n");
        printf("Options:\n");
        printf("\t--help\t display this help and exit\n");
        return 0;
    }

    /* Check for valid number of arguments */

    if (nargs > 1) {
        fprintf(stderr, "%s: too many arguments\n", args[0]);
```

```
        fprintf(stderr, "Try '%s --help' for more information\n",
                args[0]);
        return 1;
    }

    /* Print header for items from the process table */

    printf("%3s %-16s %5s %4s %4s %10s %-10s %10s\n",
           "Pid", "Name", "State", "Prio", "Ppid", "Stack Base",
           "Stack Ptr", "Stack Size");

    printf("%3s %-16s %5s %4s %4s %10s %-10s %10s\n",
           "---", "----------------", "-----", "----", "----",
           "----------", "----------", "----------");

    /* Output information for each process */

    for (i = 0; i < NPROC; i++) {
        prptr = &proctab[i];
        if (prptr-
>prstate != PR_READY) {  /* skip slots that are not ready processes   */
            continue;
        }
        printf("%3d %-16s %s %4d %4d 0x%08X 0x%08X %10d\n",
            i, prptr->prname, pstate[(int)prptr->prstate],
            prptr->prprio, prptr->prparent, prptr->prstkbase,
            prptr->prstkptr, prptr->prstklen);
    }
    return 0;
}
```

Below is the output in the terminal:



Part 2.4: xsh_wait.c

The shell command xsh_wait.c was used to create a process (user specifies the priority) and the process will print its PID before waiting on a global semaphore “globalsemaphore”.

The global semaphore is defined as “extern sid32 globalsemaphore;” in semaphore.h and is initialized in “initialize.c” as “sid32 globalsemaphore;”. Finally, it is initialized with 0 in the main.c as “globalsemaphore = semcreate(0);”

The shell command creates a process “waitglobalsem.c” which prints the PID of the newly created process and then waits on the global semaphore.

Below is xsh_wait.c:

```
/* xsh_wait.c - xsh_wait */

#include <xinu.h>
#include <stdio.h>
#include <string.h>

/*------------------------------------------------------------------------
 * xsh_wait - shell command to create a process
 *------------------------------------------------------------------------
 */
shellcmd xsh_wait(int nargs, char *args[])
{
    pid32   pid;            /* return PID           */
    pri16   priority;       /* process priority > 0     */
    char    ch;             /* next character of argument   */
    char    *chptr;         /* walks along argument string  */

    /* Check argument count */

    if (nargs > 2) {
        fprintf(stderr, "%s: incorrect argument\n", args[0]);
        fprintf(stderr, "Try '%s --help' for more information\n", args[0]);
        return SYSERR;
    }

    /* compute priority from argument string */

    if (nargs == 2) {
        chptr = args[1];
        ch = *chptr++;
        priority = 0;
        while(ch != NULLCH) {
            if ( (ch < '0') || (ch > '9') ) {
```

```
                fprintf(stderr, "%s: non-digit in process ID\n",
                    args[0]);
                return 1;
            }
            priority = 10*priority + (ch - '0');
            ch = *chptr++;
        }

        pid = create(waitglobalsem, 1024, priority, "p 1", 0);
        if (pid == SYSERR) {
            fprintf(stderr, "%s: cannot create process %d\n",
                args[0], priority);
            return 1;
        }
        resume(pid);
    }

    /* No argument so use priority default value of 20 */

    if (nargs == 1){
        pid = create(waitglobalsem, 1024, 20, "p 1", 0);
        if (pid == SYSERR) {
            fprintf(stderr, "%s: cannot create process %d\n",
                args[0], priority);
            return 1;
        }
        resume(pid);
    }
    return 0;
}
```

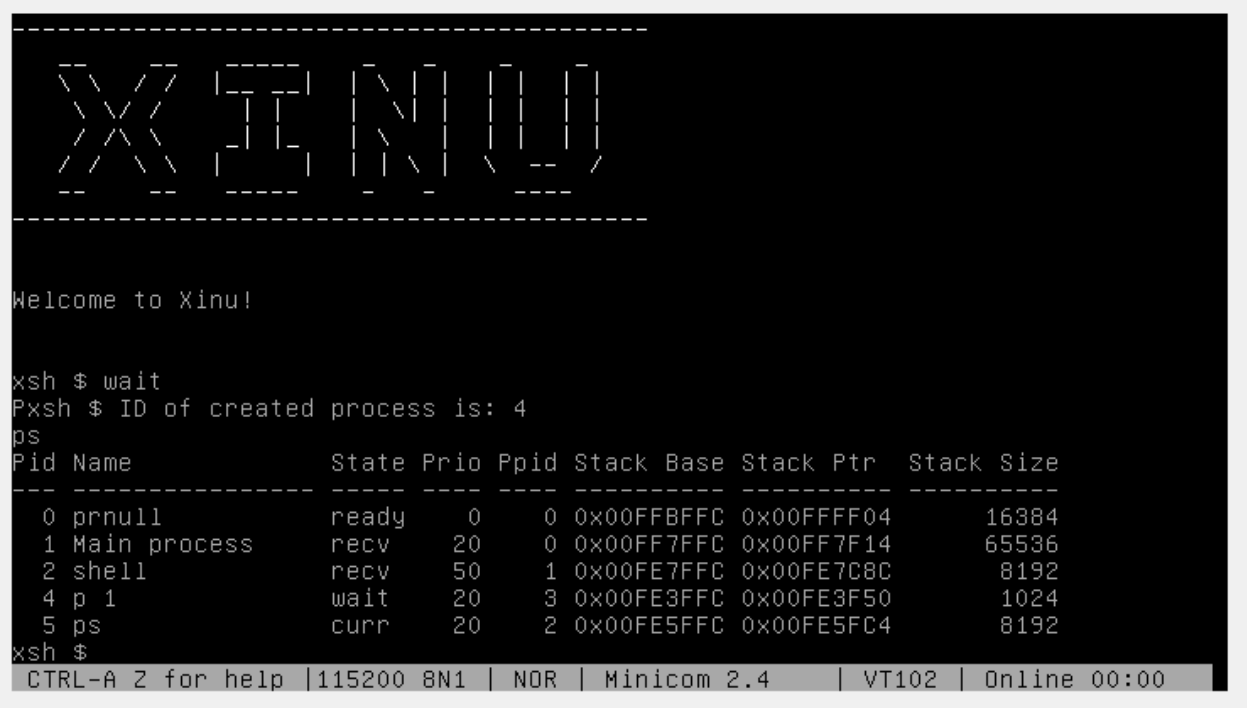
In the main.c process, the global semaphore is initialized with a count 0.

```
    globalsemaphore = semcreate(0);
```

Here is the code for waitglobalsem.c:

```
#include <xinu.h>

void waitglobalsem(void)
{
    printf("PID of created process is: %d\n", currpid);
    wait(globalsemaphore);
}
```

Below is the output in the terminal:



Part 2.5: xsh_signaln.c

The following code was used to signal n number of processes waiting on the global semaphore "globalsemaphore".

```
/* xsh_signaln.c - xsh_signaln */

#include <xinu.h>
#include <stdio.h>
#include <string.h>

/*------------------------------------------------------------------------
 * xsh_signaln - shell command to create a process
 *------------------------------------------------------------------------
 */
shellcmd xsh_signaln(int nargs, char *args[])
{
    pid32   pid;            /* return PID           */
    int32   n;      /* number of processes to signal        */
    char    ch;             /* next character of argument   */
    char    *chptr;         /* walks along argument string  */

    /* Check argument count */

    if (nargs != 2) {
        fprintf(stderr, "%s: incorrect argument\n", args[0]);
        fprintf(stderr, "Try '%s --help' for more information\n", args[0]);
        return SYSERR;
    }

    /* compute priority from argument string */

    if (nargs == 2) {
        chptr = args[1];
        ch = *chptr++;
        n = 0;
        while(ch != NULLCH) {
            if ( (ch < '0') || (ch > '9') ) {
                fprintf(stderr, "%s: non-digit in process ID\n",
                    args[0]);
                return 1;
            }
            n = 10*n + (ch - '0');
            ch = *chptr++;
        }
        signaln(globalsemaphore, n);
```

```
        //pid = create(runforever, 1024, priority, "p 1", 0);
        if (pid == SYSERR) {
            fprintf(stderr, "%s: cannot create process %d\n",
                args[0], n);
            return 1;
        }
        resume(pid);
    }
    return 0;
}
```

Below is the output on the terminal:

```
xsh $ wait
Pxsh $ ID of created process is: 4
wait
Pxsh $ ID of created process is: 6
ps
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size
--- ---------------- ----- ---- ---- ---------- ---------- ----------
  0 prnull           ready    0    0 0x00FFBFFC 0x00FFFF10      16384
  1 Main process     recv    20    0 0x00FF7FFC 0x00FF7F14      65536
  2 shell            recv    50    1 0x00FE7FFC 0x00FE7C8C       8192
  4 p 1              wait    20    3 0x00FE3FFC 0x00FE3F50       1024
  6 p 1              wait    20    5 0x00FE3BFC 0x00FE3B50       1024
  7 ps               curr    20    2 0x00FE5FFC 0x00FE5FC4       8192
xsh $ signaln 2
xsh $ ps
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size
--- ---------------- ----- ---- ---- ---------- ---------- ----------
  0 prnull           ready    0    0 0x00FFBFFC 0x00FFFF04      16384
  1 Main process     recv    20    0 0x00FF7FFC 0x00FF7F14      65536
  2 shell            recv    50    1 0x00FE7FFC 0x00FE7C8C       8192
  9 ps               curr    20    2 0x00FE5FFC 0x00FE5FC4       8192
xsh $ _
 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.4    | VT102 | Online 00:00
```

Part 2.6: xsh_resumen.c

Below is the code for xsh_resumen.c:

```
/* xsh_resumen.c - xsh_resumen */

#include <xinu.h>
#include <stdio.h>
#include <string.h>

/*------------------------------------------------------------------------
 * xsh_resumen - shell command to resume processes
 *------------------------------------------------------------------------
 */
shellcmd xsh_resumen(int nargs, char *args[])
{
    pid32   pid;        /* process priority > 0     */
    char    ch;             /* next character of argument   */
    char    *chptr;         /* walks along argument string  */
    int32   i;

    if (nargs < 2) {
        fprintf(stderr, "%s: incorrect argument\n", args[0]);
        fprintf(stderr, "Try '%s --help' for more information\n", args[0]);
        return SYSERR;
    }

    /* compute priority from argument string */
    for(i = 0; i < nargs-1; i++){
        chptr = args[1];
        ch = *chptr++;
        pid = 0;
        while(ch != NULLCH) {
            if ( (ch < '0') || (ch > '9') ) {
                fprintf(stderr, "%s: non-digit in process ID\n",
                    args[0]);
                return 1;
            }
            pid = 10*pid + (ch - '0');
            ch = *chptr++;
        }
        resume(pid);
    }
    return 0;
}
```

Here is the output in the terminal: