

# Lab 3

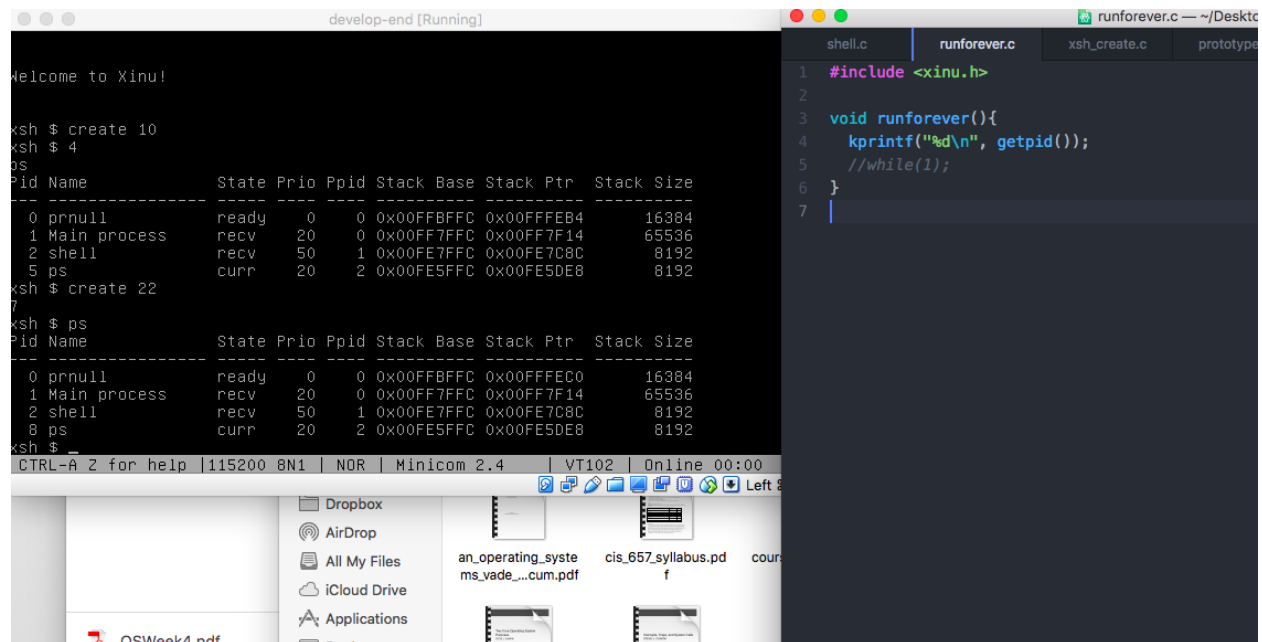
Mengqian Liu, Daniel Kalish, Jackson Taber, Kevin Martin, Lloyd Dendy

## QUESTIONS:

1. What happens if you type “create 10 or any priority less than 20”? Did you see the process ID get printed, why?

Yes you do. Create 10 generates a process with a priority of 10. At the time of running create 10, we have a main process in the receive state, with a priority of 20, and a shell process in the receive state, with a process of 50. Our create 10 process takes the current state and runs. When we execute the ps command through the terminal, the create 10 process is placed in the ready state and ps executes completely. Once ps finishes, the create 10 process resumes infinitely or until interrupted by a process of higher or equal priority.

2. What happens if you remove the “empty infinite loop” from the create shell command? Try to use the “ps” to know the answer.



The screenshot shows a terminal window titled 'develop-end [Running]' and a code editor window titled 'runforever.c'. The terminal window displays the following commands and output:

```
Welcome to Xinu!

xsh $ create 10
xsh $ 4
ps
Pid Name          State Prio Ppid Stack Base Stack Ptr Stack Size
-----
0 prnull          ready  0   0 0x00FFBFFC 0x00FFFB4 16384
1 Main process    recv  20  0 0x00FF7FFC 0x00FF7F14 65536
2 shell           recv  50  1 0x00FE7FFC 0x00FE7C8C 8192
5 ps              curr  20  2 0x00FE5FFC 0x00FE5DE8 8192

xsh $ create 22
7
xsh $ ps
Pid Name          State Prio Ppid Stack Base Stack Ptr Stack Size
-----
0 prnull          ready  0   0 0x00FFBFFC 0x00FFFB4 16384
1 Main process    recv  20  0 0x00FF7FFC 0x00FF7F14 65536
2 shell           recv  50  1 0x00FE7FFC 0x00FE7C8C 8192
8 ps              curr  20  2 0x00FE5FFC 0x00FE5DE8 8192

xsh $ _
```

The code editor window shows the following code in 'runforever.c':

```
1 #include <xinu.h>
2
3 void runforever(){
4     kprintf("%d\n", getpid());
5     //while(1);
6 }
7
```

create 10: shows the process id, 4.

Type ps, it shows the ps has the process id 5. It does not show the process id 4, because our process completed.

3. Type shell command “create 22” (note that 22 is higher than 20). What will happen in both cases below?

a. having the infinite loop in the create shell command

The screenshot shows a Minicom terminal window titled "develop-end (Running)" and a code editor window titled "runforever.c".

**Terminal Window:**

```

Welcome to Xinu!

xsh $ create 10
xsh $ 4
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	Test	ready	10	3	0x00FE3FFC	0x00FE3ED0		1024
5	ps	curr	20	2	0x00FE5FFC	0x00FE5DE8		8192

```

xsh $ create 22
7
ps

```

**Code Editor Window (runforever.c):**

```

1 #include <xinu.h>
2
3 void runforever(){
4     kprintf("%d\n", getpid());
5     while(1);
6 }
7

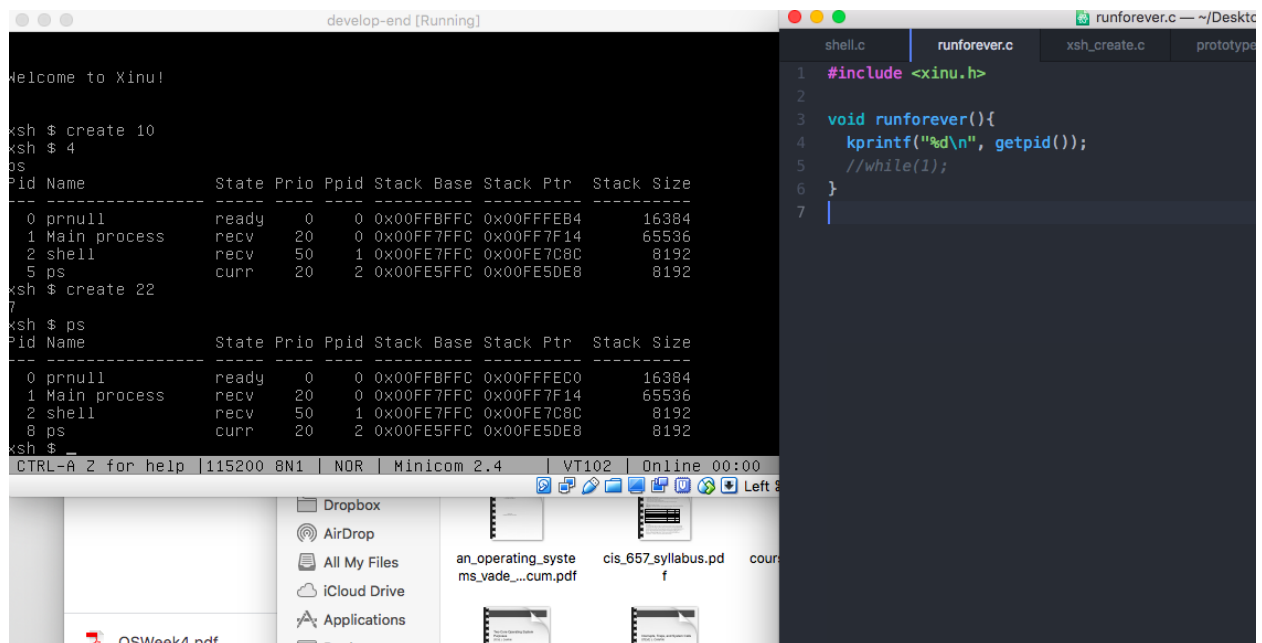
```

create 10: shows the process id, 4.

Type ps, it shows the "test" process with process id 4.

Create 22: shows the process id, 7. Executing ps shell command causes the shell (priority 50) to interrupt the infinite loop (priority 22) when receiving user input "ps," however, ps does not execute because the priority of the infinite loop is greater than the default priority 20. Since ps command's priority is not high enough, it remains in the ready queue.

b. comment out the infinite loop



```
develop-end [Running]
Welcome to Xinu!

xsh $ create 10
xsh $ 4
ps
Pid Name          State Prio Ppid Stack Base Stack Ptr Stack Size
-----
0 prnull          ready  0   0 0x00FFBFFC 0x00FFEB4 16384
1 Main process    recv   20  0 0x00FF7FFC 0x00FF7F14 65536
2 shell           recv   50  1 0x00FE7FFC 0x00FE7C8C 8192
5 ps              curr   20  2 0x00FE5FFC 0x00FE5DE8 8192

xsh $ create 22
7
xsh $ ps
Pid Name          State Prio Ppid Stack Base Stack Ptr Stack Size
-----
0 prnull          ready  0   0 0x00FFBFFC 0x00FFEC0 16384
1 Main process    recv   20  0 0x00FF7FFC 0x00FF7F14 65536
2 shell           recv   50  1 0x00FE7FFC 0x00FE7C8C 8192
8 ps              curr   20  2 0x00FE5FFC 0x00FE5DE8 8192

xsh $ _
CTRL-A 2 for help |115200 8N1 | NOR | Minicom 2.4 | VT102 | Online 00:00
```

```
runforever.c
1 #include <xinu.h>
2
3 void runforever(){
4     kprintf("%d\n", getpid());
5     //while(1);
6 }
7
```

create 10: shows the process id, 4.

Type ps, it shows the ps has the process id 5. It does not show the process id 4, because our process completes.

Create 22: shows the process id, 7. the shell command still works, because our process completed.


## FILES WE CREATED:

```
xsh_create.c x runforever.c x shell.c x prototypes.h x shprototypes.h x Makefile x
1  /* xsh_create.c - xsh_create */
2
3  #include <xinu.h>
4  #include <stdio.h>
5  #include <string.h>
6
7  /*-----
8   * xsh_create - shell command to create
9   *-----
10 */
11 shellcmd xsh_create(int nargs, char *args[])
12 {
13     pril6    priority;
14     char     ch;
15     char     *chprio;
16     pid32    pid;
17
18     if (nargs == 1) {
19         priority=INITPRIO;
20     }
21
22     else if ( nargs >= 2 ) {
23         chprio = args[1];
24         ch = *chprio++;
25         priority = 0;
26         while(ch != NULLCH) {
27             if ((ch < '0') || (ch > '9')) {
28                 kprintf("%s: non-digit in priority\n", args[1]);
29                 return 1;
30             }
31             priority = 10*priority + (ch - '0');
32             ch = *chprio++;
33         }
34
35         if (priority < (pril6)MINKEY) {
36             kprintf("%s: invalid priority\n", args[1]);
37
38             if (priority < (pril6)MINKEY) {
39                 kprintf("%s: invalid priority\n", args[1]);
40                 return 1;
41             }
42         }
43         else {
44             kprintf("Too many arguments\n");
45             return 1;
46         }
47         pid = create(runforever, 1024, priority, "Test", 0);
48
49         resume(pid);
50
51         return 0;
52
53
54 }
```



```
1  #include <xinu.h>
2
3  void runforever(){
4      kprintf("%d\n", getpid());
5      while(1);
6  }
7
```

## FILES WE CHANGED:



```
106 #-----
107 # Files for ../shell
108 #-----
109 SHELL_CFILES = \
110     addargs.c  lexan.c  shell.c
111 SHELL_CFILES += \
112     xsh_argecho.c  xsh_cat.c  xsh_clear.c  xsh_uptime.c  \
113     xsh_echo.c  xsh_exit.c  xsh_devdump.c  xsh_help.c  \
114     xsh_kill.c  xsh_memdump.c  xsh_ps.c  xsh_sleep.c  \
115     xsh_memstat.c  xsh_create.c
116 SHELL_CFULL = ${SHELL_CFILES:%=../shell/%}
117 SRC_FILES += $(SHELL_CFULL)
118 #####
119 # Generate a list of all object files
120 #####
121 OBJ_TMP = $(patsubst %.c,%.o,${SRC_FILES}) # substitute .c => .o
122 OBJ_FILES = $(patsubst %.S,%.o,${OBJ_TMP}) # substitute .S => .o
123
124 # Export variables for recursive make calls (such as the library)
125 export
```

```

xsh_create.c xsh_runforever.c shell.c prototypes.h shprototypes.h Makefile
46
47 #-----
48 # Files for ../system #
49 #-----
50
51 SYSTEM_SFILES = \
52     start.S      ctxsw.S      clkint.S      intr.S
53
54 SYSTEM_CFILES = \
55     asctime.c    bufinit.c    chprio.c    panic.c    \
56     clkinit.c    close.c      conf.c      control.c   \
57     create.c     freebuf.c    freemem.c  getbuf.c    \
58     getc.c       getdev.c     getitem.c  getmem.c    \
59     getpid.c     getprio.c    getstk.c   initialize.c \
60     i386.c       insert.c     insertd.c  ioerr.c     \
61     ionull.c     kill.c        kprintf.c  main.c      \
62     mkbufpool.c  newqueue.c  open.c     pci.c       \
63     putc.c       queue.c      read.c     ready.c     \
64     receive.c    recvclr.c    recvtime.c resched.c   \
65     resume.c     sched_cntl.c seek.c      semcount.c  \
66     semcreate.c  semdelete.c semreset.c send.c      \
67     signal.c     signaln.c    sleep.c    suspend.c   \
68     unsleep.c    userret.c    wait.c     wakeup.c    \
69     write.c      xdone.c      yield.c    evect.c     runforever.c
70
71 SYSTEM_SFULL = ${SYSTEM_SFILES:%=../system/%}
72 SYSTEM_CFULL = ${SYSTEM_CFILES:%=../system/%}
73
74 SRC_FILES += $(SYSTEM_SFULL)
75 SRC_FILES += $(SYSTEM_CFULL)
76

```

```

xsh_create.c xsh_runforever.c shell.c prototypes.h shprototypes.h Makefile
1  /* shell.c - shell */
2
3  #include <xinu.h>
4  #include <stdio.h>
5  #include "shprototypes.h"
6
7  /*-----
8   * Xinu shell commands and the function associated with each
9   *-----*/
10
11 const struct cmdent cmdtab[] = {
12     {"argecho", TRUE, xsh_argecho},
13     {"cat", FALSE, xsh_cat},
14     {"clear", TRUE, xsh_clear},
15     {"devdump", FALSE, xsh_devdump},
16     {"echo", FALSE, xsh_echo},
17     {"exit", TRUE, xsh_exit},
18     {"help", FALSE, xsh_help},
19     {"kill", TRUE, xsh_kill},
20     {"memdump", FALSE, xsh_memdump},
21     {"memstat", FALSE, xsh_memstat},
22     {"ps", FALSE, xsh_ps},
23     {"sleep", FALSE, xsh_sleep},
24     {"?", FALSE, xsh_help},
25     {"create", FALSE, xsh_create}
26 };
27
28 uint32 ncmd = sizeof(cmdtab) / sizeof(struct cmdent);
29

```



```
xsh_create.c x runforever.c x shell.c x prototypes.h x shprototypes.h x Makefile x
1  /* in file runforever.c */
2  extern void runforever();
3
4
5
6
7  /* in file addargs.c */
8  extern status addargs(pid32, int32, int32[], int32, char *, void *);
9
10 /* in file ascdatetime.c */
11 extern status ascdatetime(uint32, char *);
12
13 /* in file bufinit.c */
14 extern status bufinit(void);
15
16 /* in file chprio.c */
17 extern pril6 chprio(pid32, pril6);
18
```

```
xsh_create.c x runforever.c x shell.c x prototypes.h x shprototypes.h x Makefile x
39
40 /* in file xsh_memdump.c */
41 extern shellcmd xsh_memdump (int32, char *[]);
42
43 /* in file xsh_memstat.c */
44 extern shellcmd xsh_memstat (int32, char *[]);
45
46 /* in file xsh_nvram.c */
47 extern shellcmd xsh_nvram (int32, char *[]);
48
49 /* in file xsh_ping.c */
50 extern shellcmd xsh_ping (int32, char *[]);
51
52 /* in file xsh_ps.c */
53 extern shellcmd xsh_ps (int32, char *[]);
54
55 /* in file xsh_sleep.c */
56 extern shellcmd xsh_sleep (int32, char *[]);
57
58 /* in file xsh_udpdump.c */
59 extern shellcmd xsh_udpdump (int32, char *[]);
60
61 /* in file xsh_udpecho.c */
62 extern shellcmd xsh_udpecho (int32, char *[]);
63
64 /* in file xsh_udpserver.c */
65 extern shellcmd xsh_udpserver (int32, char *[]);
66
67 /* in file xsh_uptime.c */
68 extern shellcmd xsh_uptime (int32, char *[]);
69
70 /* in file xsh_help.c */
71 extern shellcmd xsh_help (int32, char *[]);
72
73 /* in file xsh_create.c */
74 extern shellcmd xsh_create (int32, char *[]);
75
```