Lab 3

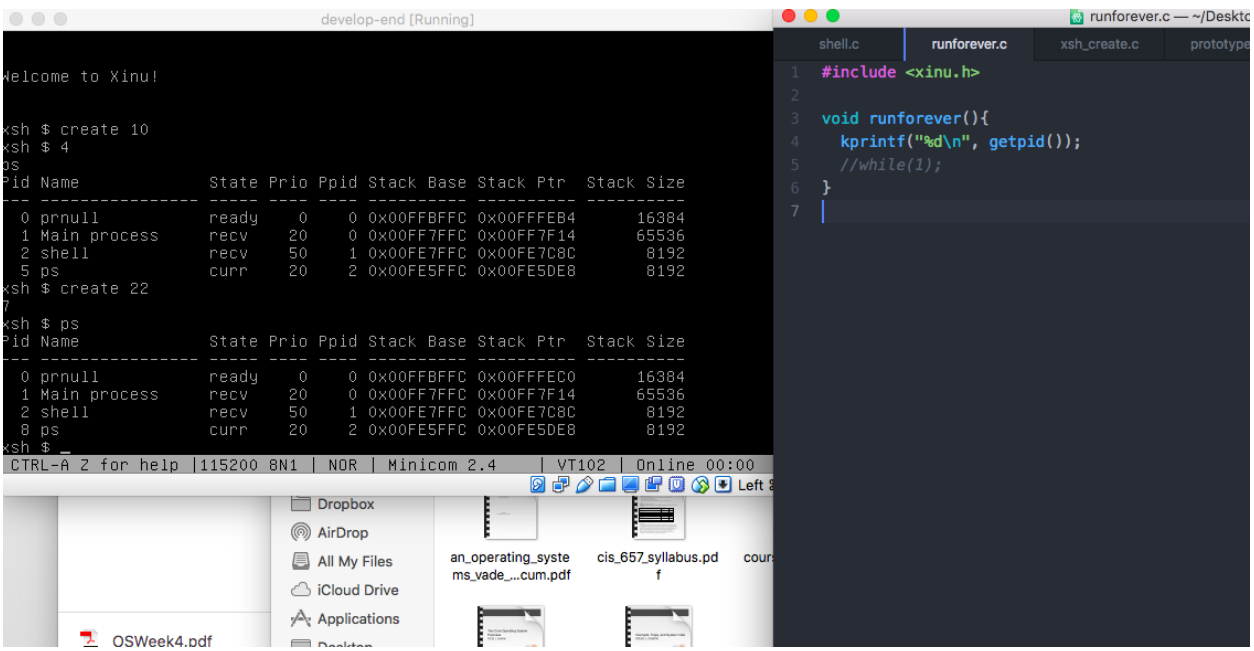
Mengqian Liu, Daniel Kalish, Jackson Taber, Kevin Martin, Lloyd Dendy

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.

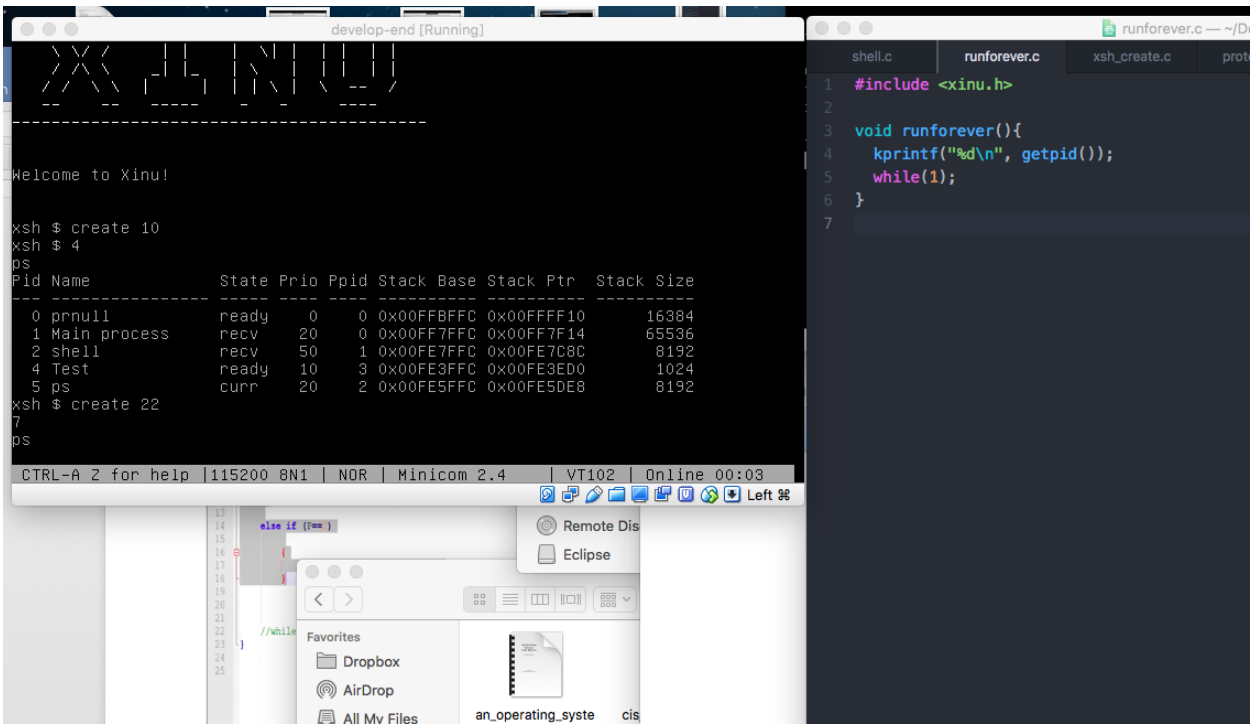


create 10: shows the process id, 4.

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

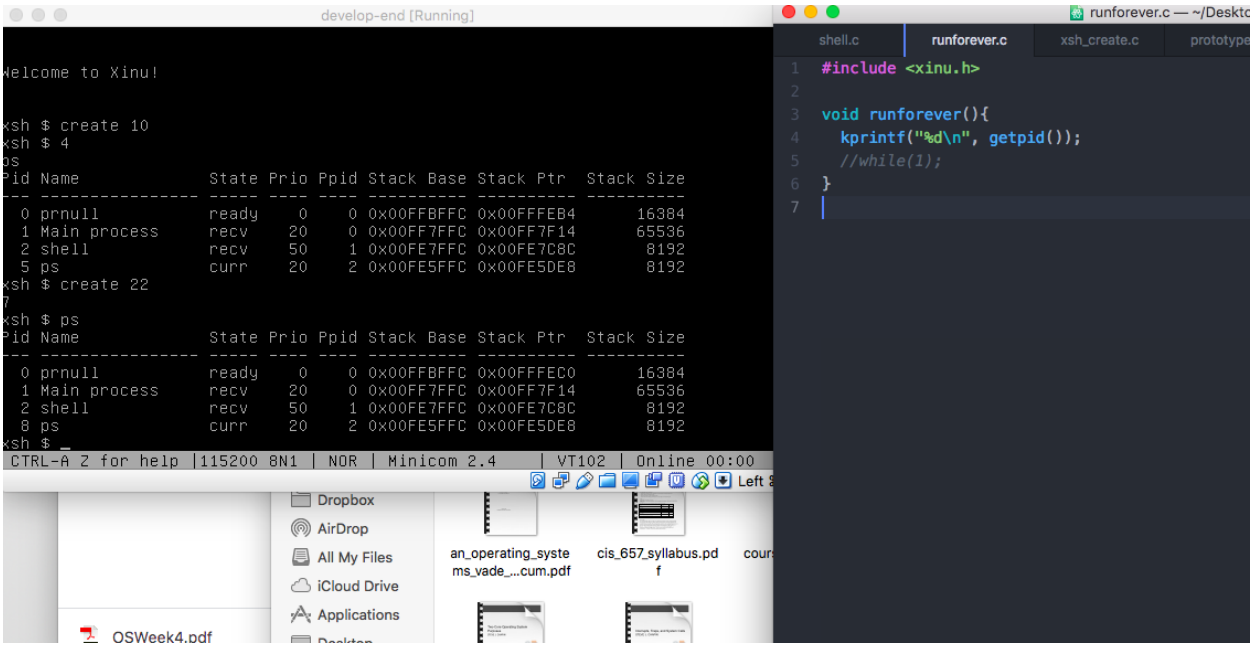


create 10: shows the process id, 4.

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



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 | runforever.c | shell.c | prototypes.h | shprototypes.h | Makefile

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

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

/*------------------------------------------------------------------------
 * xsh_create - shell command to create
 *------------------------------------------------------------------------
 */
shellcmd xsh_create(int nargs, char *args[])
{
    pri16   priority;
    char    ch;
    char    *chprio;
    pid32   pid;

    if (nargs == 1) {
        priority=INITPRIO;

    }

    else if ( nargs >= 2 ) {
        chprio = args[1];
        ch = *chprio++;
        priority = 0;
        while(ch != NULLCH) {
            if ((ch <'0') || (ch > '9')) {
                kprintf("%s: non-digit in priority\n", args[1]);
                return 1;
            }
            priority = 10*priority + (ch - '0');
            ch = *chprio++;
        }

        if (priority < (pri16)MINKEY) {
            kprintf("%s: invalid priority\n", args[1]);
```


```c
        if (priority < (pri16)MINKEY) {
            kprintf("%s: invalid priority\n", args[1]);
```


```c
            return 1;
        }

    }
    else {
        kprintf("Too many arguments\n");
        return 1;
    }
    pid = create(runforever, 1024, priority, "Test", 0);

    resume(pid);

    return 0;



}
```

xsh_create.c | runforever.c | shell.c | prototypes.h | shprototypes.h | Makefile

```c
#include <xinu.h>

void runforever(){
	kprintf("%d\n", getpid());
	while(1);
}

```

FILES WE CHANGED:

xsh_create.c | runforever.c | shell.c | prototypes.h | shprototypes.h | Makefile

```
#-----------------------------------------------------------------------
# Files for ../shell
#-----------------------------------------------------------------------

SHELL_CFILES =                                  \
    addargs.c   lexan.c     shell.c

SHELL_CFILES +=                                 \
    xsh_argecho.c   xsh_cat.c   xsh_clear.c     xsh_uptime.c    \
    xsh_echo.c  xsh_exit.c  xsh_devdump.c   xsh_help.c  \
    xsh_kill.c  xsh_memdump.c   xsh_ps.c    xsh_sleep.c     \
    xsh_memstat.c   xsh_create.c

SHELL_CFULL  = ${SHELL_CFILES:%=../shell/%}

SRC_FILES += $(SHELL_CFULL)

########################################
# Generate a list of all object files
########################################

OBJ_TMP = $(patsubst %.c,%.o,$(SRC_FILES))  # substitute .c => .o
OBJ_FILES  = $(patsubst %.S,%.o,$(OBJ_TMP))     # substitute .S => .o


# Export variables for recursive make calls (such as the library)
export
```

xsh_create.c | runforever.c | shell.c | prototypes.h | shprototypes.h | Makefile

```makefile

#-----------------------------------------------------------------------
# Files for ../system                                   #
#-----------------------------------------------------------------------

SYSTEM_SFILES =                                 \
    start.S     ctxsw.S     clkint.S    intr.S

SYSTEM_CFILES =                                 \
    ascdate.c   bufinit.c   chprio.c    panic.c     \
    clkinit.c   close.c     conf.c      control.c   \
    create.c    freebuf.c   freemem.c   getbuf.c    \
    getc.c      getdev.c    getitem.c   getmem.c    \
    getpid.c    getprio.c   getstk.c    initialize.c    \
    i386.c      insert.c    insertd.c   ioerr.c     \
    ionull.c    kill.c      kprintf.c   main.c      \
    mkbufpool.c newqueue.c  open.c      pci.c       \
    putc.c      queue.c     read.c      ready.c     \
    receive.c   recvclr.c   recvtime.c  resched.c   \
    resume.c    sched_cntl.c    seek.c      semcount.c  \
    semcreate.c semdelete.c semreset.c  send.c      \
    signal.c    signaln.c   sleep.c     suspend.c   \
    unsleep.c   userret.c   wait.c      wakeup.c    \
    write.c     xdone.c     yield.c     evec.c runforever.c

SYSTEM_SFULL = ${SYSTEM_SFILES:%=../system/%}
SYSTEM_CFULL = ${SYSTEM_CFILES:%=../system/%}

SRC_FILES += $(SYSTEM_SFULL)
SRC_FILES += $(SYSTEM_CFULL)

```

xsh_create.c | runforever.c | shell.c | prototypes.h | shprototypes.h | Makefile

```c
/* shell.c  -  shell */

#include <xinu.h>
#include <stdio.h>
#include "shprototypes.h"

/************************************************************************/
/* Xinu shell commands and the function associated with each        */
/************************************************************************/
const   struct  cmdent  cmdtab[] = {
    {"argecho", TRUE,   xsh_argecho},
    {"cat",     FALSE,  xsh_cat},
    {"clear",   TRUE,   xsh_clear},
    {"devdump", FALSE,  xsh_devdump},
    {"echo",    FALSE,  xsh_echo},
    {"exit",    TRUE,   xsh_exit},
    {"help",    FALSE,  xsh_help},
    {"kill",    TRUE,   xsh_kill},
    {"memdump", FALSE,  xsh_memdump},
    {"memstat", FALSE,  xsh_memstat},
    {"ps",      FALSE,  xsh_ps},
    {"sleep",   FALSE,  xsh_sleep},
    {"?",       FALSE,  xsh_help},
    {"create", FALSE,   xsh_create}

};

uint32  ncmd = sizeof(cmdtab) / sizeof(struct cmdent);
```

xsh_create.c | runforever.c | shell.c | prototypes.h | shprototypes.h | Makefile

```c
/* in file runforever.c */
extern  void runforever();




/* in file addargs.c */
extern  status  addargs(pid32, int32, int32[], int32,char *, void *);

/* in file ascdate.c */
extern  status  ascdate(uint32, char *);

/* in file bufinit.c */
extern  status  bufinit(void);

/* in file chprio.c */
extern  pri16   chprio(pid32, pri16);
```

xsh_create.c | runforever.c | shell.c | prototypes.h | shprototypes.h | Makefile

```c

/* in file xsh_memdump.c */
extern  shellcmd  xsh_memdump   (int32, char *[]);

/* in file xsh_memstat.c */
extern  shellcmd  xsh_memstat   (int32, char *[]);

/* in file xsh_nvram.c */
extern  shellcmd  xsh_nvram (int32, char *[]);

/* in file xsh_ping.c */
extern  shellcmd  xsh_ping  (int32, char *[]);

/* in file xsh_ps.c */
extern  shellcmd  xsh_ps    (int32, char *[]);

/* in file xsh_sleep.c */
extern  shellcmd  xsh_sleep (int32, char *[]);

/* in file xsh_udpdump.c */
extern  shellcmd  xsh_udpdump   (int32, char *[]);

/* in file xsh_udpecho.c */
extern  shellcmd  xsh_udpecho   (int32, char *[]);

/* in file xsh_udpeserver.c */
extern  shellcmd  xsh_udpeserver (int32, char *[]);

/* in file xsh_uptime.c */
extern  shellcmd  xsh_uptime    (int32, char *[]);

/* in file xsh_help.c */
extern  shellcmd  xsh_help  (int32, char *[]);

/* in file xsh_create.c */
extern  shellcmd  xsh_create    (int32, char *[]);

```