```
[000.000104] 00000000000i[      ] BXSHARE not set. using compile time default '/usr/local/share/bochs'
[000.000420] 00000000000i[      ] reading configuration from /tmp/tmp.dtuJqGM4tj
[000.000612] 00000000000i[      ] installing term module as the Bochs GUI
[000.000950] 00000000000i[      ] using log file /output/bochsout.student.txt
[000.001267] ========================================================================
[000.001494]                         Bochs x86 Emulator 2.6.8
[000.001715]                Built from SVN snapshot on May 3, 2015
[000.001847]                  Compiled on Oct  3 2016 at 05:23:39
[000.002036] ========================================================================
[007.035356] (B)0[1;24r[m[?7h[?1h=[H[J[H[JOptions: apmbios pcibios pnpbios eltorito rombios32
[007.035481]
[007.035651]
[007.035952] Press F12 for boot menu.
[007.036237]
[007.036436] Booting from Floppy...
[007.036749]
[007.036896] Loading........
[007.036996]
[007.037252]
[007.037493] CPSC 415, 2016W1
[007.037665] 32 Bit Xeros 0.01
[007.037773] Located at: 0 to 8a90
[007.037890] Some sample output to illustrate different types of printing
[007.038043]
[007.038103]
[007.038162]
[007.038261] The 1024string is: "This is the number -69 when printed signed -69 unsigned 4294[19;1H967227 hex ffffffbb and a string Hello.
[007.038349] [33DSample printing of 1024 in signed 1024, unsigned 1024 and hex 400."
[007.038405]
[007.038476] The formula is 1024 + -69 = 955.
[007.038531]
[011.598096] [47C
[011.598287]
[011.598591]
[011.598868]
[011.599150]
[011.599581] [4AThis is the number -69 when printed signed -69 unsigned 4294967227 hex ffffffbb
[011.599956] and a string Hello.
[011.600351] [13DSample printing of 1024 in signed 1024, unsigned 1024 and hex 400.
[011.600712]
[059.828873] Welcome to Xeros - an experimental OS
```

```
1  /* user.c : User processes
2  */
3
4  #include <xeroskernel.h>
5  #include <xeroslib.h>
6
7  #define BUF_MAX 100
8  char *username = "cs415\n";
9  char *password = "EveryoneGetsAnA\n";
10 char *pscom = "ps";
11 char *excom = "ex";
12 char *kcom = "k";
13 char *acom = "a";
14 char *tcom = "t";
15 char *psand = "ps&";
16 char *exand = "ex&";
17 char *kand = "k&";
18 char *aand = "a&";
19 char *tand = "t&";
20 int shellPid;
21 int alarmTicks;
22
23
24 void shell(void);
25 void  root(void);
26 void psf(void);
27 void exf(void);
28 void kf(int pid);
29 void alarmHandler(void *cntx);
30 void alarm(void);
31 void t(void);
32 int parseString(char *inBuf, int inBufSize, char *outBuf, int outBufSize);
33
34 void  root( void ) {
35     int error = 0;
36     char ubuf[BUF_MAX];
37     char pbuf[BUF_MAX];
38
39     while (1) {
40         // Banner
41         sysputs("\nWelcome to Xeros - an experimental OS\n");
42
43         // Open keyboard in non echo mode
44         int fd = sysopen(0);
45         if (fd == -1) {
46             kprintf("Error opening keyboard\n");
47             for(;;);
48         }
49
50         // Turn keyboard echoing on;
51         error = sysioctl(fd, 56);
52         if (error == -1) {
53             kprintf("Error turning keyboard echoing on\n");
54             for(;;);
55         }
56
57         sysputs("Username: ");
58         int bytes = sysread(fd, &ubuf[0], BUF_MAX - 1);
59         if (!bytes) {
60             kprintf("Sysread returned EOF\n");
61             for(;;);
62         }
63         if (bytes == -1) {
```

```
64              kprintf("Sysread returned an error\n");
65              for(;;);
66          }
67          ubuf[bytes] = NULLCH;
68          // Turn keyboard echoing off;
69          error = sysioctl(fd, 55);
70          if (error == -1) {
71              kprintf("Error turning keyboard echoing off\n");
72              for(;;);
73          }
74          sysputs("Password: ");
75          bytes = sysread(fd, &pbuf[0], BUF_MAX - 1);
76          if (!bytes) {
77              kprintf("Sysread returned EOF\n");
78              for(;;);
79          }
80          if (bytes == -1) {
81              kprintf("Sysread returned an error\n");
82              for(;;);
83          }
84          pbuf[bytes] = NULLCH;
85          error = sysclose(fd); // Just writing this in for testing even though we
     dont actually have to close the fd
86          if (error == -1) {
87              kprintf("Error turning closing device\n");
88              for(;;);
89          }
90          int usercheck = strcmp(&ubuf[0], username);
91          int passcheck = strcmp(&pbuf[0], password);
92          //kprintf("\n user check %d, pass check %d", usercheck, passcheck);
93          //kprintf("\n user in %s, pass in %s", ubuf, pbuf);
94          if (usercheck == 0 && passcheck == 0) {
95              break;
96          }
97      }
98      char buf[BUF_MAX];
99      sprintf(buf, "\n");
100     sysputs(buf);
101
102     shellPid = create(&shell, 8000);
103     int retCode = syswait(shellPid);
104     sprintf(buf, "Syswait retcode%d\n", retCode);
105     sysputs(buf);
106
107 }
108
109 void shell(void) {
110     char stdinput[BUF_MAX];
111     // Open keyboard in echo mode
112     int fd = sysopen(1);
113     if (fd == -1) {
114         kprintf("Error opening keyboard\n");
115         for(;;);
116     }
117
118     while (1) {
119         sysputs("> ");
120         int bytes = sysread(fd, &stdinput[0], BUF_MAX - 1);
121         if (!bytes) {
122             break;
123         }
124         if (bytes == -1) {
125             kprintf("Sysread returned an error\n");
```

```
126             for(;;);
127         }
128         stdinput[bytes++] = NULLCH;
129         char command[bytes];
130         int bytesParsed = parseString(stdinput, bytes, command, bytes);
131         if (bytesParsed == -2) {
132             // GO back to the the beginning of the loop
133             sysputs("Ignoring command");
134             continue;
135         }
136         command[bytesParsed++] = NULLCH;
137         if (!strcmp(command, pscom) || !strcmp(command, psand)) {
138                 psf();
139         } else if (!strcmp(command, excom) || !strcmp(command, exand)) {
140             break;
141         } else if (!strcmp(command, kcom) || !strcmp(command, kand)) {
142             if (bytesParsed < BUF_MAX) {
143                 char arg[BUF_MAX];
144                 bytesParsed += parseString(&stdinput[bytesParsed], BUF_MAX - byt
    esParsed, arg, BUF_MAX);
145                 arg[bytesParsed++] = NULLCH;
146                 int pid = atoi(arg);
147                 int res = syskill(pid, 9);
148                 if (res == -712) {
149                     sprintf(arg, "No such process\n");
150                     sysputs(arg);
151                 }
152             }
153         } else if (!strcmp(command, acom) || !strcmp(command, aand)) {
154             if (bytesParsed < BUF_MAX) {
155                 char arg[BUF_MAX];
156                 bytesParsed += parseString(&stdinput[bytesParsed], BUF_MAX - byt
    esParsed, arg, BUF_MAX);
157                 arg[bytesParsed++] = NULLCH;
158                 int ticks = atoi(arg);
159                 alarmTicks = ticks;
160                 syssighandler(15, &alarmHandler, NULL);
161                 int alarmPid = syscreate(&alarm, 8000);
162                 if (!strcmp(command, acom)) {
163                     syswait(alarmPid);
164                 }
165             }
166         } else if (!strcmp(command, tcom) || !strcmp(command, tand)) {
167             int tpid = syscreate(&t, 8000);
168             if (!strcmp(command, tcom)) {
169                 syswait(tpid);
170             }
171         } else {
172             sysputs("Command not found\n");
173         }
174     }
175     sysputs("Exiting shell...\n");
176 }
177
178 int parseString(char *inBuf, int inBufSize, char *outBuf, int outBufSize) {
179         int bytesRead = 0;
180         char * endInBuf = inBuf + inBufSize;
181         char * endOutBuf = outBuf + outBufSize;
182         while (inBuf < endInBuf && *inBuf == ' ') {
183             inBuf++;
184         }
185         while (inBuf < endInBuf && *inBuf != ' ' && *inBuf != '\n' && outBuf < en
    dOutBuf ) {
```

```
186                 *outBuf = *inBuf;
187                 outBuf++;
188                 inBuf++;
189                 bytesRead++;
190             }
191             return bytesRead;
192     }
193
194     void psf(void) {
195         struct processStatuses ps;
196         int procs =  sysgetcputimes(&ps);
197         char buf[100];
198         sprintf(buf, "%4s  %4s  %10s\n", "Pid", "Status", "CpuTime");
199         sysputs(buf);
200         for (int i = 0; i <= procs; i++) {
201             int status = ps.status[i];
202             switch(status) {
203             case STATE_STOPPED:
204                 sprintf(buf, "%4d  %4s  %10d\n", ps.pid[i], "STOPPED", ps.cpuTime[i])
    ;
205                 break;
206             case STATE_READY:
207                 sprintf(buf, "%4d  %4s  %10d\n", ps.pid[i], "READY", ps.cpuTime[i]);
208                 break;
209             case STATE_SLEEP:
210                 sprintf(buf, "%4d  %4s  %10d\n", ps.pid[i], "SLEEP", ps.cpuTime[i]);
211                 break;
212             case STATE_RUNNING:
213                 sprintf(buf, "%4d  %4s  %10d\n", ps.pid[i], "RUNNING", ps.cpuTime[i]
    );
214                 break;
215             case STATE_RECV:
216                 sprintf(buf, "%4d  %4s  %10d\n", ps.pid[i], "RECV", ps.cpuTime[i]);
217                 break;
218             case STATE_SEND:
219                 sprintf(buf, "%4d  %4s  %10d\n", ps.pid[i], "SENDING", ps.cpuTime[i])
    ;
220                 break;
221             case STATE_WAITING:
222                 sprintf(buf, "%4d  %4s  %10d\n", ps.pid[i], "WAITING", ps.cpuTime[i])
    ;
223                 break;
224             case STATE_DEV_WAITING:
225                 sprintf(buf, "%4d  %4s  %10d\n", ps.pid[i], "DEV_WAITING", ps.cpuTim
    e[i]);
226                 break;
227             }
228             sysputs(buf);
229         }
230     }
231     void kf(int pid) {}
232
233     void alarmHandler(void *cntx) {
234         char buf[100];
235         sprintf(buf, "ALARM ALARM ALARM\n");
236         sysputs(buf);
237         syssighandler(15, NULL, NULL);
238     }
239
240     void alarm(void) {
241         int sleepTime = alarmTicks * TICKLENGTH;
242         syssighandler(9, &sysstop, NULL);
243         syssleep(sleepTime);
```

```
244     syskill(shellPid, 15);
245 }
246
247 void t(void) {
248     char buf[5];
249     syssighandler(9, &sysstop, NULL);
250     sprintf(buf, "T\n");
251     for (;;) {
252         syssleep(10000);
253         sysputs(buf);
254     }
255 }
```

```
+ compile_code
+ local user_c
+ [[ 0 -gt 0 ]]
+ user_c=
+ local EXTRA_SRC
+ EXTRA_SRC=("$@")
+ local INTERRUPTS=1
+ [[ -z '' ]]
+ header 'COMPILATION (user.c=<student>, interrupts=1)'
+ set +x
********************
COMPILATION (user.c=<student>, interrupts=1)
********************

+ set -x
+ cd /xeros
+ git clean -f
+ find . -name '*.o' -o -name '*.la' -o -name '*.a' -exec rm -v -- '{}' ';'
removed './lib/libxc.a'
removed './lib/libxc/libxc.a'
+ [[ -z '' ]]
+ [[ 1 -eq 0 ]]
+ make clean
cd compile; make clean
make[1]: Entering directory '/xeros/compile'
rm -rf *.o *.bak *.a core errs ./xeros ./xeros.boot
make[1]: Leaving directory '/xeros/compile'
cd boot; make clean
make[1]: Entering directory '/xeros/boot'
rm -f zImage
rm -f zBoot/zSystem.out
make -C zBoot clean
make[2]: Entering directory '/xeros/boot/zBoot'
rm -f *.o
rm -f zSystem
make[2]: Leaving directory '/xeros/boot/zBoot'
make[1]: Leaving directory '/xeros/boot'
rm -f bochsout.txt
+ make
cd compile; make
make[1]: Entering directory '/xeros/compile'
gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -E -DLINUX -I../h -DLOCORE -DSTANDALONE -DAT386 -DBRELOC=0x100000 -DBOOTPLOC=0x150000 -DLINUX_XINU ../c/startup.S | as --32 -o startup.o
gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -E -DLINUX -I../h -DLOCORE -DSTANDALONE -DAT386 ../c/intr.S | as --32 -o intr.o
gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prototypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/`basename init.o .o`.[c]
gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prototypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/`basename i386.o .o`.[c]
gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prototypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/`basename evec.o .o`.[c]
gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prototypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/`basename kprintf.o .o`.[c]
gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prototypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/`basename mem.o .o`.[c]
gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prototypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/`basename dis
```

```
    p.o .o‘.[c]
50  gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prot
    otypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/‘basename cts
    w.o .o‘.[c]
51  ../c/ctsw.c:15:14: warning: ’k_stack’ defined but not used [-Wunused-variable]
52   static void *k_stack;
53                ^
54  gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prot
    otypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/‘basename sys
    call.o .o‘.[c]
55  gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prot
    otypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/‘basename cre
    ate.o .o‘.[c]
56  gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prot
    otypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/‘basename use
    r.o .o‘.[c]
57  gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prot
    otypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/‘basename msg
    .o .o‘.[c]
58  gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prot
    otypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/‘basename sle
    ep.o .o‘.[c]
59  gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prot
    otypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/‘basename sig
    nal.o .o‘.[c]
60  gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prot
    otypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/‘basename kbd
    .o .o‘.[c]
61  gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__  -Wall -Wstrict-prot
    otypes -fno-builtin -c  -DBSDURG  -DVERBOSE -DPRINTERR -I../h ../c/‘basename di_
    calls.o .o‘.[c]
62  (cd ../lib/libxc; make install)
63  make[2]: Entering directory ’/xeros/lib/libxc’
64  rm -f libxc.a
65  ar cr libxc.a abs.o atof.o atoi.o atol.o blkcopy.o ctype_.o doprnt.o doscan.o ec
    vt.o fgets.o fprintf.o fputs.o gets.o index.o printf.o puts.o qsort.o rand.o rin
    dex.o scanf.o   sprintf.o strcat.o strcmp.o strcpy.o strlen.o strncat.o strncmp.
    o strncpy.o swab.o memset.o
66  cp libxc.a ../libxc.a
67  make[2]: Leaving directory ’/xeros/lib/libxc’
68  ld -m elf_i386 -e start -Ttext 0x000000 startup.o intr.o  init.o i386.o evec.o k
    printf.o mem.o disp.o ctsw.o syscall.o create.o user.o msg.o sleep.o signal.o kb
    d.o di_calls.o  ../lib/libxc.a -o ./xeros
69  make[1]: Leaving directory ’/xeros/compile’
70  cd boot; make
71  make[1]: Entering directory ’/xeros/boot’
72  make -C zBoot zSystem
73  make[2]: Entering directory ’/xeros/boot/zBoot’
74  rm -f tmppiggy tmppiggy.gz tmppiggy.lnk; \
75  objcopy  -O binary -R .note -R .comment -R .stab -R .stabstr ../../compile/xeros
     tmppiggy; \
76  gzip -f -9 < tmppiggy > tmppiggy.gz; \
77  echo "SECTIONS { .data : { input_len = .; LONG(input_data_end - input_data) inpu
    t_data = .; *(.data) input_data_end = .; }}" > tmppiggy.lnk; \
78    ld -m elf_i386 -m elf_i386 -r -o piggy.o -b binary tmppiggy.gz -b elf32-i386 -
    T tmppiggy.lnk; \
79  rm -f tmppiggy tmppiggy.gz tmppiggy.lnk
80  gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__ -traditional -c -I/u
    sr/include head.S
81  gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__ -fno-builtin -O2 -DS
    TDC_HEADERS  -c -o inflate.o inflate.c
82  gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__ -fno-builtin -O2 -DS
    TDC_HEADERS  -c -o unzip.o unzip.c
```

```
gcc -m32 -march=i386 -std=gnu99 -D__KERNEL__ -D__ASSEMBLY__ -fno-builtin -O2 -DSTDC_HEADERS  -c -o misc.o misc.c
ld -m elf_i386 -Ttext 0x1000 -e startup_32 -o zSystem  head.o inflate.o unzip.o misc.o piggy.o
make[2]: Leaving directory '/xeros/boot/zBoot'
gcc -m32 -march=i386  -std=gnu99 -O2 -fomit-frame-pointer -D__BFD__ -I/usr/include -o build build.c
objcopy  -O binary -R .note -R .comment -R .stab -R .stabstr zBoot/zSystem zBoot/zSystem.out; \
./build boot/bootsect boot/setup zBoot/zSystem.out CURRENT > zImage
Root device is (0, 42)
Boot sector 512 bytes.
Setup is 1980 bytes.
System is 28 kB
make[1]: Leaving directory '/xeros/boot'
+ echo 0
```