



ICT1012

OPERATING SYSTEMS

LABORATORY MANUAL

Lab1-w1: Lab Assignments

AY 2025/26

BACHELOR OF ENGINEERING
IN
INFORMATION AND COMMUNICATIONS TECHNOLOGY

Contents

Lab Assignment	3
Assignment Task 1: sleep.....	3
Assignment Task 2: memory dump.....	3
Assignment submission.....	5

Lab Assignment

Assignment Task 1: sleep

- Implement a user-level sleep program for xv6, like that of the UNIX sleep command. “sleep” should pause for a user-specified number of ticks.
- A tick is a quantum of time defined by the xv6 kernel, namely the time between two interrupts from the timer chip.
- Place your code in “*user/sleep.c*”.
- Refer to other programs in “*user*” folder (e.g., “*user/echo.c*”, “*user/grep.c*”, and “*user/rm.c*”) to see how command-line arguments are passed to a program.
- Add your sleep program to “*UPROGS*” in “*makefile*”
- Execute “*make qemu*” to compile your program and you’ll be able to run it from the xv6 shell.
- If the user forgets to pass an argument, sleep should print an error message.
- The command-line argument is passed as a string. Convert it to an integer using “*atoi*” (refer to “*user/ulib.c*”).
- Use the system call “*pause*”.
- See “*kernel/sysproc.c*” for the xv6 kernel code that implements the “*pause()*” system call (look for “*sys_pause*”), “*user/user.h*” for the C definition of “*pause()*” callable from a user program, and “*user/usys.S*” for the assembler code that jumps from user code into the kernel for “*pause()*”.
- *sleep*’s main should call “*exit(0)*” when it is done.
- Run the program from the xv6 shell:

```
:/mnt/d/ICT1012//xv6labs-w1$ make qemu
...
init: starting sh
$ sleep 10
(nothing happens for a little while)
$
```

- Exit xv6 shell with keys “**Ctrl + a**” followed by “**x**”
- Run all test cases for “*sleep*”

```
:/mnt/d/ICT1012//xv6labs-w1$ ./grade-lab-util sleep
make: 'kernel/kernel' is up to date.
== Test sleep, no arguments == sleep, no arguments: OK (9.9s)
== Test sleep, returns == sleep, returns: OK (9.1s)
== Test sleep, makes syscall == sleep, makes syscall: OK (6.3s)
```

Assignment Task 2: memory dump

- Use the template file “*user/memdump.c*”.
- Implement the function “*memdump(char *fmt, char *data)*”.
- Purpose of “*memdump()*” is to print the contents of the memory pointed to by “*data*” in the format described by the “*fmt*” argument.
- The format is a C string. Each character of the string indicates how to print successive parts of the data.
- “*memdump()*” should handle the following format characters:
 - i: print the next 4 bytes of the data as a 32-bit integer, in decimal.
 - p: print the next 8 bytes of the data as a 64-bit integer, in hex.

- h: print the next 2 bytes of the data as a 16-bit integer, in decimal.
- c: print the next 1 byte of the data as an 8-bit ASCII character.
- s: the next 8 bytes of the data contain a 64-bit pointer to a C string; print the string.
- S: the rest of the data contains the bytes of a null-terminated C string; print the string.
- Use C's “`printf()`” in “`memdump()`”.
- The “`memdump`” program, if executed with no arguments, calls “`memdump()`” with some example format strings and data. If “`memdump()`” is correctly implemented, the output will be:

```
$ memdump
```

Example 1:

61810

2025

Example 2:

a string

Example 3:

another

Example 4:

BD0

1819438967

100

z

xyzzy

Example 5:

hello

w

o

r

l

d

Note: Example 4 output may show different hex address.

- If the “`memdump`” program is invoked with an argument, it will read its standard input up to an end of file and then call “`memdump()`” with the format and input data. So, once “`memdump()`” is implemented:

```
$ echo deadc0de | memdump hhcccc
```

25956

25697

c

0

d

e

```
$ echo deadc0de | memdump p
```

6564306364616564

Correct output

- Exit xv6 shell with keys “**Ctrl + a**” followed by “x”
- Run all test cases for “`memdump`”

```
$ ./grade-lab-util memdump
```

make: 'kernel/kernel' is up to date.

== Test memdump, examples == memdump, examples: OK (13.4s)

== Test memdump, format ii, S, p == memdump, format ii, S, p: OK (9.1s)

Assignment submission

- Execute “**make grade**” to test both “sleep” and “memdump” tasks

```
$ make grade
make clean

...
== Test sleep, no arguments ==
$ make qemu-gdb
sleep, no arguments: OK (60.0s)
== Test sleep, returns ==
$ make qemu-gdb
sleep, returns: OK (6.5s)
== Test sleep, makes syscall ==
$ make qemu-gdb
sleep, makes syscall: OK (7.9s)
== Test memdump, examples ==
$ make qemu-gdb
memdump, examples: OK (6.7s)
== Test memdump, format ii, S, p ==
$ make qemu-gdb
memdump, format ii, S, p: OK (7.0s)
Score: 40/40
```

- Execute “**make zipball**” to create “*lab.zip*”

```
$ make zipball
...
Created lab.zip
```

- Upload the following to xSiTe ICT1012 Dropbox folder “xv6labs-w1: Utils”
 - lab.zip
 - Screenshot of results of “make grade”