



ICT1012

OPERATING SYSTEMS

LABORATORY MANUAL

Lab1-w2: Lab Assignments

AY 2025/26

BACHELOR OF ENGINEERING
IN
INFORMATION AND COMMUNICATIONS TECHNOLOGY

Contents

Lab Assignment	3
Prerequisite	3
Assignment Task 1: hello	3
Assignment Task 2: sixfive	5
Assignment Task 3: xargs	6
Assignment submission.....	7

Lab Assignment

Prerequisite

- Download “**lab.zip**” from “xSiTe” folder “**Labs/xv6labs-w2: Utilities**” to “**c:/ICT1012/xv6labs-w2**”.
- Unzip and keep all folders and files directly under “**c:/ICT1012/xv6labs-w2**” without the “labs” sub-folder.

Assignment Task 1: hello

- Your solution should be in the file “*user/hello.c*”.
- In this task, we create a new system call to use in “xv6”.
- Add a simple “*print*” system call – “*int hello(void);*”, return 0 on success.
- Kernel handler “*uint64 sys_hello(void)*” prints a fixed line “*Hello from kernel syscall!*” and then a customised line (“*I am ICT1012*”) using “*printf*” (safe from kernel context)
- In “*kernel/syscall.h*”, add a unique “*syscall*” number after existing ones. Numbers are to be kept contiguous to avoid collision.

```
#define SYS_mkdir  20
#define SYS_close  21
#define SYS_interpose  22
#define SYS_hello 23
```

- In “*kernel/syscall.c*”, add the “*extern*” declaration and table entry.

```
extern uint64 sys_link(void);
extern uint64 sys_mkdir(void);
extern uint64 sys_close(void);
extern uint64 sys_hello(void);
```

```
[SYS_unlink] sys_unlink,
[SYS_link]   sys_link,
[SYS_mkdir]  sys_mkdir,
[SYS_close]  sys_close,
[SYS_hello] sys_hello,
};
```

- In “*kernel/sysproc.c*”, add code for the simple “*syscall*”

```
uint64
sys_hello(void)
{
    printf("Hello from kernel syscall!\n");
    printf("I am ICT1012!\n");
    return 0;
}
```

- In “*kernel/defs.h*”, add export prototype to allow other kernel files reference it,

```
// syscall.c
void      argint(int, int*);
int       argstr(int, char*, int);
void      argaddr(int, uint64 *);
int       fetchstr(uint64, char*, int);
int       fetchaddr(uint64, uint64*);
void      syscall();
uint64    sys_hello(void);
```

- In “user/usys.pl”, add an entry so that the build system generates RISC-V assembly stub in “usys.S” to load register “a7” with “SYS_hello”, executes “ecall” and returns result in register “a0”.

```
entry("getpid");
entry("sbrk");
entry("pause");
entry("uptime");
entry("hello");
```

- In “user/user.h”, add codes for the simple “syscall”

```
int getpid(void);
char* sys_sbrk(int, int);
int pause(int);
int uptime(void);
int hello(void);
```

- Create “user/hello.c” to test the new system call.

```
// user/hello.c
#include "kernel/types.h"
#include "user/user.h"

int
main(void)
{
    int r = hello();
    printf("hello returned %d\n", r);
    return 0;
}
```

- Edit “Makefile” to add to “UPROGS”

```
UPROGS=\
    $U/_forphan\
    $U/_dorphan\
    $U/_hello\
```

- Rebuild “xv6”. Recommended to “**make clean**” when kernel files or “Makefile” is modified to regenerate “fs.img”

```
$ make clean && make qemu
```

- Run “hello” in the shell

```
xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$ hello
Hello from kernel syscall!
I am ICT1012!
hello returned 0
$
```

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

```
:/mnt/c/ICT1012/xv6labs-w2$ ./grade-lab-util hello
make: 'kernel/kernel' is up to date.
== Test hello == hello: OK (1.5s)
:/mnt/c/ICT1012/xv6labs-w2$
```

Assignment Task 2: sixfive

- Your solution should be in the file “*user/sixfive.c*”.
- For this task, use the system calls “*open*” and “*read*”, “*C strings*”, and processing text files in C.
- For each input file, “*sixfive*” must print all the numbers in the file that are multiples of 5 or 6.
- Numbers are a sequence of decimal digits separated by characters in the string “*-\r\t\n.,*”.
- For the six in “xv6”, “*sixfive*” shouldn't print 6 but, for example, “/6,” it should.
- Hints:
 - Read the input file a character at the time
 - Test if a character match any of the separators using “*strchr*” (see “*user/ulib.c*”).
 - Start and end of file are implicit separators.
 - Add the program to “*UPROGS*” in “*Makefile*”.
 - Changes to the file system persist across runs of “*qemu*”; to get a clean file system run “**make clean**” and then “**make qemu**”.
- The following example illustrates “*sixfive*’s” behavior:

```
$ sixfive sixfive.txt
5
100
18
6
$
```

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

```
:/mnt/c/ICT1012/xv6labs-w2$ ./grade-lab-util sixfive
make: 'kernel/kernel' is up to date.
== Test sixfive_test == sixfive_test: OK (2.5s)
== Test sixfive_readme == sixfive_readme: OK (1.7s)
```

```
== Test sixfive_all == sixfive_all: OK (1.4s)
:/mnt/c/ICT1012/xv6labs-w2$
```

Assignment Task 3: xargs

- Your solution should be in the file “*user/xargs.c*”.
- Write a simple version of the UNIX “*xargs*” program for xv6.
- Its arguments describe a command to run, it reads lines from the standard input, and it runs the command for each line, appending the line to the command's arguments.
- Hints:
 - Use “*fork*” and “*exec*” to invoke the command on each line of input. Use “*wait*” in the parent to wait for the child to complete the command.
 - To read individual lines of input, read a character at a time until a newline (“\n”) appears.
 - “*kernel/param.h*” declares “*MAXARG*”, which may be useful if you need to declare an “*argv*” array.
 - Add the program to “*UPROGS*” in “*Makefile*”.
 - Changes to the file system persist across runs of “*qemu*”; to get a clean file system run “**make clean**” and then “**make qemu**”.
- The following example illustrates “*xarg*’s” behaviour:

```
$ echo hello too | xargs echo bye
bye hello too
$
```

- Note that the command here is “*echo bye*” and the additional arguments are “*hello too*”, making the command “*echo bye hello too*”, which outputs “*bye hello too*”.
- Please note that “*xargs*” on UNIX makes an optimization where it will feed more than one argument to the command at a time.
- We don't expect you to make this optimization. To make “*xargs*” on UNIX behave the way we want it to for this lab, please run it with the “*-n*” option set to 1.
- For instance,

```
$ (echo 1 ; echo 2) | xargs -n 1 echo
1
2
$
```

- To test your solution for “*xargs*”, run the shell script “**xargstest.sh**”.
- Your solution should produce the following output:

```
$
$ sh < xargstest.sh
$ $ $ $ $ $ hello
hello
hello
$ $
```

- The output has many \$ because the xv6 shell doesn't realize it is processing commands from a file instead of from the console, and prints a \$ for each command in the file.
- Exit xv6 shell with keys “**Ctrl + a**” followed by “**x**”.
- Run all test cases for “xargs”.

```
:/mnt/c/ICT1012/xv6labs-w2$ ./grade-lab-util xargs
make: 'kernel/kernel' is up to date.
== Test xargs == xargs: OK (2.8s)
== Test xargs, multi-line echo == xargs, multi-line echo: OK (0.9s)
:/mnt/c/ICT1012/xv6labs-w2$
```

Assignment submission

1. Execute “**make grade**” to test “*hello*”, “*sixfive*” and “*xargs*” tasks.

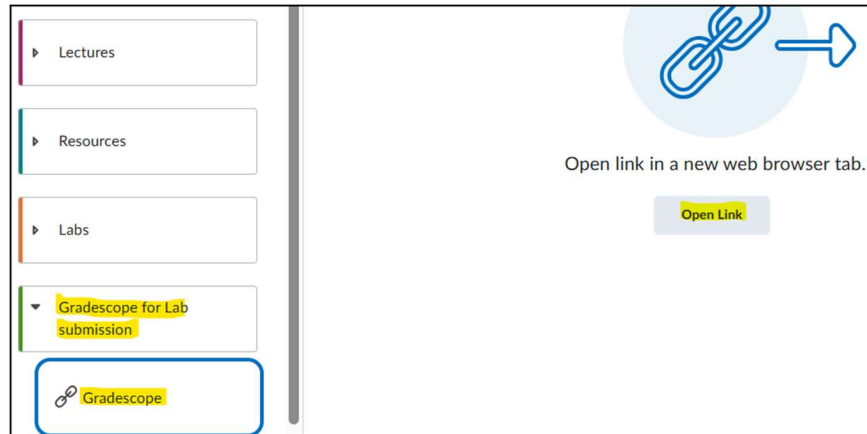
```
:/mnt/c/ICT1012/xv6labs-w2$ make grade
make clean
. . .

make[1]: Leaving directory '/mnt/c/ICT1012/xv6labs-w2'
== Test hello ==
$ make qemu-gdb
hello: OK (11.0s)
== Test sixfive_test ==
$ make qemu-gdb
sixfive_test: OK (0.8s)
== Test sixfive_readme ==
$ make qemu-gdb
sixfive_readme: OK (1.5s)
== Test sixfive_all ==
$ make qemu-gdb
sixfive_all: OK (1.5s)
== Test xargs ==
$ make qemu-gdb
xargs: OK (1.8s)
== Test xargs, multi-line echo ==
$ make qemu-gdb
xargs, multi-line echo: OK (1.2s)
Score: 55/55
:/mnt/c/ICT1012/xv6labs-w2$
```

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

```
:/mnt/c/ICT1012/xv6labs-w2$ make zipball
...
Created lab.zip
```

3. Upload the “**lab.zip**” file to Gradescope to get auto graded.
 - In xSiTe, navigate to “Gradescope for Lab submission”. Click the link, “Gradescope”.



- Login with SIT credentials.
- Select assignment “**xv6labs-w2: Utilities**”. Submit “**lab.zip**” that you had generated in step 2.

The image shows a screenshot of the Gradescope website. The browser address bar shows 'gradescope.com/courses/1207688/assignments'. The page title is 'SIT-2520-ICT1012' and the course description is 'ICT1012-Operating Systems (2025/26 T2)'. The page shows a list of assignments under the heading '2 Assignments'.

Name	Points	Released	Due (+08)
xv6labs-w1: Utilities	40.0	JAN 6, 2026 11:00 AM	JAN 16, 2026 11:59 PM
xv6labs-w2: Utilities	55.0	JAN 15, 2026 8:00 AM	JAN 21, 2026 11:59 PM