

ICT1012

OPERATING SYSTEMS

LABORATORY INSTRUCTIONS

Quiz 2 – Question 1

Contents

Lab extension task (4 marks)	2
Prerequisite:	2
Task: Word-Level Handshake (4 Bytes).....	2
Task Requirements	2
Task Testing.....	4
Quiz Submission.....	4

Lab extension task (4 marks)

Prerequisite:

Download the “**quiz2-q1.zip**” file from “LMS/xSiTe” folder:

“**Lab Quizzes (Prerequisites) / Quiz 2 / Question 1**”

Task: Word-Level Handshake (4 Bytes)

In lab 1 week 3, you have implemented a user-level program “user/handshake.c” that transfers **one byte** between a parent and child process using two pipes.

In this extended task, you are required to modify the given incomplete/missing code file “**user/handshake.c**” to support the transfer of **one word (4 bytes)** instead of a single byte. This code will not compile because there are place holders that need to be corrected with required statement(s).

Task Requirements

1. Modify the program so that:
 - a) The parent process sends a **4-byte word** to the child process.
 - Shall the input parameter to handshake be a string with less than 4 characters (hence it is not a 4-byte word) the parent will fill the missing characters with the byte 0.
 - Shall the input parameter to handshake be a string with more than 4 characters, the parent will truncate the input to 4 characters.
 - b) The child:
 - Reads exactly 4 bytes from the parent
 - Prints
 <pid>: child received <word> from parent
 - Sends the same 4-byte word back to the parent
 - c) The parent:
 - Reads exactly 4 bytes from the child
 - Prints
 <pid>: parent received <word> from child

- d) Both processes must properly close unused pipe ends.
- e) The program must use only xv6 system calls:
 - o pipe
 - o fork
 - o read
 - o write
 - o close
 - o wait
 - o getpid
 - o exit

2. Add it to the “**Makefile**” to compile and grade the task.

3. Program Behaviour

Example execution:

```
$ handshake 1234
5: child received 1234 from parent
4: parent received 1234 from child
$
```

Where:

- "1234" represents a 4-byte word
- The exact PIDs may vary

Example execution:

```
$ handshake 12
5: child received 12 from parent
4: parent received 12 from child
```

Example execution:

```
$ handshake 12345
5: child received 1234 from parent
4: parent received 1234 from child
```

Task Testing

1. Changes to the file system persist across runs of “*qemu*”; to get a clean file system run “**make clean && make qemu**”. The following example illustrates “*handshake*’s” behaviour in “*qemu*”:

```
xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$ handshake 1012
4: child received 1012 from parent
3: parent received 1012 from child
$
```

2. Testing with the grading script “**grade-quiz**”:

```
$ ./grade-quiz handshake
== Test handshake == handshake: OK (1.7s)
$
```

3. Testing with the “**make grade**”

```
$ make clean && make grade
== Test handshake ==
$ make qemu-gdb
handshake: OK (6.7s)
Score: 1/1
$
```

NOTE: The grade script does not cover all the possibilities and it is your responsibility to carry out further testing

Quiz Submission

1. After “**make grade**” and “**make zipball**”, submit the “**lab.zip**” file to Gradescope Assignment “**xv6labs-quiz2-q1-handshake**” for autograding.
Note: Create the zip file with “**make zipball**” and not with Windows zip program or iOS Files App to avoid zipping multiple hierarchical directories.
2. Autograder will execute the same “*make grade*” when you submit your “*lab.zip*” file. This is for submission verification only. The score will not be included in the total score.
3. Test case script may not cover all the possibilities and it is your responsibility to carry out further testing.

4. After the quiz due date and time, all submissions will be regraded again with different set of test cases.
5. To ensure your submission goes through without technical issues, please submit at the earliest.