CS 3423 Operating Systems
Fall Semester 2017
Prof. Pai H. Chou

Assignment 3

Original Due Date: Sunday, October 8, 2017, 11:59pm
=> **Revised Due Date**: *October 15, 2017, 11:59pm.*
*** Don't forget about Assignment 4, which is also due on October 15 at the same time! ***
Submit electronically on iLMS

What to submit: One zip file named <studentID>-`hw3.zip` (replace <studentID> with your own student ID). It should contain four files:

- one PDF file named **hw3.pdf** for Section 1 and Section 2. Write your answers in **English**. Check your spelling and grammar. Include your name and student ID!
- Section 2: turn in your modified `exception.cc` and the typescript file named `typescript2` to show the build and run session.
- Section 3: Python source file named **hw3.py** (exact upper and lower case) and the typescript file named `typescript3` on the telnet side.

1. [40 points] Problem Set

From Chapter 3 of Sieberchatz 9th edition book. "Processes"

1. [20 points] 3.2: Describe the actions taken by a kernel to context-switch between processes.
2. [20 points] 3.11.a: What are the benefits and the disadvantages of each of the following? Consider both the system level and the programmer level.
   a. Synchronous and asynchronous communication

2. [40 points] Nachos Exercise

This is a continuation of last week's `Write(char *buffer, int size, OpenFileId id)` system call. Be sure you are able to understand last week's assignment completely, or else you won't be able to do this part. If you could not get your code working for hw2, contact a TA to get graded for hw2 first and then obtain a working copy of the relevant files. Last week, you were asked to use DEBUG statement to print the parameter values passed to the system call and the return value from the system call to the user program.

Modify the {userprog}`/exception.cc:ExceptionHandler()` code from hw2 by typecasting the first parameter value (which corresponds to the buffer pointer) to `(char*)` type and use `DEBUG` statement to print it as a string. However, it does not print "Hello world\n", and in fact it most likely results in a core dump (crash).

1. [20 points] Explain why dereferencing the first parameter in `ExceptionHandler()` does not result in the string whose address is passed by the statement `Write("Hello world\n", 12, 1);` in {test}`/hw2.c:main()`.

2. [20 points] Explain the correct way for `ExceptionHandler()` to obtain the string, by giving both verbal description and the C code to do it. Hint: the answer can be found easily in other system call examples in the same code. Otherwise, study the `Machine` class in {machine}`/machine.h` and find the relevant data structure. Then, modify `exception.cc` to implement this change. Remove the DEBUG statements that you added for hw2, and add a DEBUG statement to print the string. Turn in the modified exception.cc file, and also submit a typescript that shows how you rebuild Nachos (i.e., make clean; make in build.linux directory) and running the same hw2 executable.

3. [20 points] Python Programming

This week, you are to write code for a producer-consumer problem using message passing.

Python supports pretty much all of the system programming libraries as C does, but in a slightly different (and convenient) way. For sockets programming, you can use the module named `socket`. You can get help in interactive mode:

```
$ python3
>>> import socket
>>> help(socket)
Help on module socket:

NAME
    socket
...
```

A socket is an endpoint of a point-to-point connection over TCP/IP network between two processes on either the same machine or different machines. There are two phases: (1) establishing a connection (2) sending and receiving data.

3.1 Establishing a Connection

Connections need to be established by two processes that wish to communicate with each other. The side making a request to connect is the *client*, and the side awaiting connection is called the *server*. Your assignment is to write the server side; you will use the `telnet` program as the client.

To make a connection, you can use the following sample code.

```
import socket
def MakeServerSocket(host='', port=8888, limit=10):
     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
     try:
          s.bind((host, port))
     except socket.error as msg:
```

```
            print('bind fails: '+str(msg[0]))
            import sys
            sys.exit(-1)
    s.listen(limit)
    return s
```

The return value from calling
`s = MakeServerSocket()`
will be a “server socket” ready for accepting connection requests from clients. Any potential client can send request to the server’s IP address at the given port number (defaults to 8888 in this case), and up to *limit* (defaults to 10) clients may have pending requests.

For the server to accept connection requests, it should do
`conn, addr = s.accept()`
This is a blocking call. It will not return until a client comes and requests a connection. if successful, then `conn` contains the actual socket for the server to communicate with the client, and `addr` is the IP address of the client. Note that the difference between `s` and `conn` is that s is for accepting connections only but cannot be used for communication, whereas conn is for actual sending and receiving.

3.2 Sending and Receiving

The methods to use for communication are send() and recv(). In the example below, `conn` is just the name of the socket object from `accept`.

- `conn.send(`*data*`)` ; you may want to use `conn.sendall()` instead if you want to make sure all bytes are sent.
- `conn.recv(`*numOfBytes*`)`

In Python, you can’t just send a `str` type (string), because str is more general and can be encoded differently. So, you have to use the encode method to convert the string to an array of bytes. For example, if you want to send the string `'hello'`, you need to pass `'hello'.encode('utf-8')`, since utf-8 is a way to encode Unicode using 8-bit (namely byte) sequences.

For this assignment, you are to write a simple server program in Python to return the string of one value at a time from the `YieldBST()` function you wrote for hw2. Your server needs to look like this:

- create a server socket (using the code above)
- use the server socket to accept client requests and get the socket for communicating with the client.
- make the generator object by calling `YieldBST(T)`; you may call it `gen`.
- go into a loop to iterate over `gen`
    - receive from client but discard the message
    - convert the item from the generator (which is an `int`) to a `str` and then to a byte-like array using the `.encode()` method above
    - send it to the client

You don't have to write a client program; instead, you can use `telnet`, which is a common text-mode client program found in most Unix systems. To run both the client and server on the same machine, you may open two terminal windows. In one, you start the server by typing

```
$ python3 hw3.py
```

Assuming you use the default port number, then in another window, you type

```
$ telnet localhost 8888
```

Then you should be able to talk to the server by typing. You may also print additional messages to display the status of your program.

For example, a telnet session may look like this:

```
$ telnet 127.0.0.1 8888
Connected to localhost.
Escape character is '^]'.
> hello  # this is just junk ignored by the server
6
> world  # you can type and delete; client won't send till enter
12
> a      # more stuff, or just enter
14
> b
17
> cdefg
32
> hijklm
35
> opq
40
Connection closed by foreign host.
$
```

Use the script program to capture the session when running telnet, similar to the example above, but rename the typescript file as `typescript3` before submitting.