Amirkabir University of Technology

Computer engineering and it department
Operating Systems

Project 1

Authors:
Amir Hossein Rassafi (97123191)
Mohammad Navid Shahsavari (9229023)

Prof. Nastooh Taheri Javan

March 15, 2019

Question 1

Clone the source code of the xv6 operating system from the Git.

Find the PCB of this Operating System and answer the following questions:

1. What components is the structure related to pcb made of?
2. Explain the purpose of storing each of the following variables in pcb.
    - Sz
    - State
    - Context
    - Ofile
    - Killed

Question 2

In this part you should add a **system call** named **getppid** to xv6 operating system that returns pid of the parent of current process.
you can find more details of the xv6 operating system here

Question 3

Choose one of the following problems and implement it. The aim of both problems is to sort an array with a fixed size of 10000 using fork. In both problems array is defined as a shared memory.

1. Implement **Merge-sort** using fork, which means that, at each step, after dividing the array into two parts, sorting of each part should be done in parallel and using fork.
2. Implement **Quick-sort** using fork, which means that, after choosing the pivot, sorting of the remaining two parts should be done in parallel and using fork.

If the specific part of the array had M(or less) number of elements,there's no need to fork.(Go on as it is said in the question).

- Run your program for different values of M. Does your run-time increase for the very small values of M? If yes, explain the reason.

You can use this link to see sample codes of shared memory.

Question 4

The objective of this assignment is to practice socket programming by writing a chat application. The assignment consists of implementing two components: a chat client and a chat server. The server maintains information about the clients that have registered with a specific chat group and dispatches the messages sent by the clients. Clients can communicate with others by registering with a group. All received messages (from other clients) should be sent to standard output. The server and clients should be invoked with:

- Sever: server [server port number]
- Client: client [server host name] [server port number] [client name]

After successfully contacting the server, the client should support the following commands:

- join [groupID]: registers the client with a group, allowing it to receive and send messages from/to the specified group.
- send [groupID] [message]: allows a client to send a message to a group.
- leave [groupID]: indicates that a client wants to stop sending and receiving messages.
- quit: closes the client application

When a client joins a group, the server registers the client with the corresponding group. A client can be member of multiple groups simultaneously and duplicate join requests must be discarded. Similarly, when a client requests to leave a group, the server removes the client from the corresponding group; if a client tries to leave a group in which it is not a member, the request must be discarded. When the server receives a send request message for a group, the server delivers the message to all members of the group The message format should be as follows: "ClientName: msg".

Hint: You should use select() in the server to perform multiplexing among the clients (i.e., no busy wait). select() allows a program to monitor multiple file descriptors (socket descriptor in our case), waiting until one or more of them become ready