CS377: Assignment 2, 16 January 2015
Processes, Signals and Shared Memory

A process P creates child processes $Q_1 \ . \ . \ . \ Q_n$, where the first child process to be created is called Q_1, the next one is Q_2, etc., and fulfills some data processing requirements of the user by using the child processes.
The general idea is that we need to store email addresses, however, each domain (eg gmail.com, outlook.com) is handled by a particular child process Q_i. The process P is responsible for overall coordination while each child process is responsible for a particular domain and stores all the user IDs corresponding to that domain. These can be maintained in an array like structure and each entry has an index. You may assume the user IDs, domain names and extensions to contain alphanumeric characters, without special characters. The only special characters @ and . demarcate the three components in an email id, e.g., xyz@gmail.com. accepts commands from a user and fulfills them by
P accepts commands from a user and fulfills them by

- Sending instructions to a child process by sending it signals, and
- Sending data to a child process through shared memory.

A child process manages the data items handed to it by P through shared memory and responds to P s queries also by using shared memory:

P must respond to the following commands from the user:

1. *add_email abc@xyz.com*: This command should check if the domain xyz.com already exists. If not, P first creates a child process for this domain. P then sends this email address to the corresponding child (Q_i) via shared memory. Q_i then checks if the email address already exists. If so, Q_i puts the following message in shared memory:
   `Child process xyz.com - Email address already exists.`
   else Q puts the message:
   `Child process xyz.com - Email address abc@xyz.com added successfully.`
   and sends a signal to P to indicate that it has put a message in shared memory and is ready to receive instructions in the form of signals. (Q_i should use the signal SIGUSR1.)

2. *search_email abc@xyz.com*: P should print an error message as below if the domain xyz.com is not created yet.
   `Parent process - Domain does not exist.`
   Otherwise, P sends the email address to the corresponding child process (Q_i) via shared memory. Q_i searches for the email address and sends a signal to P indicating whether the email address exists or not. It also should send the corresponding index of this user ID (abc) in shared memory (essentially index of its position in an array if an array is used)
   P upon receiving the signal, should print one of the following messages appropriately.
   `Parent process - could not find the email address abc@xyz.com`
   or
   `Parent process - found the email address abc@xyz.com at <index>`

3. *delete_email abc@xyz.com*: P should print an error message as below if the domain xyz.com is not created yet.
   `Parent process - Domain does not exist.`
   Otherwise, P sends the email address to the corresponding child process (Q_i) via shared memory. Q_i searches for the email address and deletes it. Additionally it replies to P with the index of the deleted element using shared memory.
   P then prints
   `Parent process - child xyz.com could not find the email address abc@xyz.com`
   or
   `Child process - child xyz.com deleted abc@xyz.com from position <index>.`

4. *delete_domain xyz.com*: P should print an error message as below if the domain xyz.com is not created yet or has already been deleted.
   `Parent process - Domain does not exist.`
   Otherwise, P then kills the child process (Q_i) corresponding to the domain xyz.com and displays its PID the following.
   `Parent process - Domain xyz.com with PID - <PID>deleted.`
   Before exiting Q_i prints all the user IDs stored by it.

5. *send_email abc@sender.com xyz@receiver.com message_body*:
   The parent process, P, must print the following message if either sender or receiver does not exist:
   `Parent process - The sender/receiver does not exist`
   If both receiver & sender exists, then P must send the message body along with email of the receiver and the username of the sender (in this case ”abc”) to the sender process. The sender process should now use these details to send the message to the receiver process. After sending the message, it should print:
   `Child Process - Email sent from abc@sender.com to xyz@receiver.com`
   After receiving the email, the receiver process should respond by printing the following message:
   `Child Process - Email received at process <PID>for user xyz`

Submission: Submit a single tar file "RollNo1_RollNo2.tar.gz" per group by 5 pm. The tar file must contain a README file (giving names, roll numbers of the group members and the instructions to run the code) and C/C++ files.

Note

- An additional document *help.pdf* is also attached. Take a look at it, it contains some relevant functions along with their syntax and description. It will help you in the assignment. For more information use the man pages.
- P should keep a list of all domains & Q_is should keep list of email ids(which belongs to domain of Q_i)
- You will be able to do the assignment using a single shared memory area