Systems Software/Programming – Lab Manual

Lab 7 – Inter Process Communication using Pipe

Write C program for each of the problem below. After executing each of the C program, you will need to capture the output in text file format.
C program file will be named as StudentID_Lab7_x.c
Text file with output captured will be names as StudentID_Lab7_x.txt

Problem 1: Child sends a specific line from a file to parent

- Create parent and child process as StudentID_Lab7_1_parent.c and StudentID_Lab7_1_child.c respectively
  - The program StudentID_Lab7_1_parent.c will take two command line arguments. First command line argument is filename and 2nd argument is line number. For example

    ```
    $ ./ StudentID_Lab7_1_parent.out test.txt 5
    ```
  - Parent process uses fork() to create a child process
  - Before calling exec() to run the child process, parent process uses dup2() system call to use standard output file descriptor for child process as pipe file descriptor.
  - It then calls exec() system call, while also supplying the command line arguments to this child program (as StudentID_Lab7_1_child.c). It supplies both filename and line number to child process.
  - Child process reads the content of the file passed in 1st command line argument and gets the string from line number specified in 2nd argument (e.g. 5th line from test.txt)
  - Child process then displays the string using printf() but it will be send to pipe file descriptor by OS to parent process (because of dup2), which parent process will display. After sending string with child process closes the open file handles and exits
  - Parent process after receiving string from pipe and displaying it, it waits for child to exit using wait() system call.
  - As soon as parent child process is finished, parent will come out of wait() and it also exits.

Problem 2: Q&A using Parent-Child (Create parent and child process as StudentID_Lab7_2.c).

- Consider you have two variables for questions and answers as below in both parent and child processes.
  char* questions[] = {"quit", "You study in which university?", "which course are you studying?", "what is your area of interest?"};
  char* answers[] = {"quit", "DAIICT", "Systems Programming", "Embedded Systems"};
- Parent process will accept input as question number from user. If the question number is between 0 to 3, the actual text of question will be sent from parent to child process using → write(fd1[WRITE], questions[que], strlen(questions[que])+1);
- Once parent sends the question to child, it just waits for getting answer back from child using → bytesRead= read(fd2[READ], message, MSGLEN);
- When child process receives the question using → bytesRead= read(fd1[READ], message, MSGLEN);
- Child process then find the index of the question that it received from questions[] string array. It uses that index to get the correct answer from the answers[] string array
- Child process then replies back with an answer to the parent using → write(fd2[WRITE], answers[que], strlen(answers[que])+1);
- If child finds that the question number is equal to 0 (i.e. quit) then it first sends string "quit" as an answer to parent and then it will exit after closing all the open file handles. Parent process upon reading question number equal to 0 and after receiving "quit" as the answer from child, closes all open file handles and waits for child to exit using wait() system call to ensure that we don't create a zambie child process. Once parent comes out of wait() system call, it also exits.

Optional Assignment: As you can see Problem 1 can be extended to read multiple files in several child processes at the same time and reply to parent with a requested line number in separate pipes. You can implement this version to used multiple child processes using different pipes.

Submission: StudentID_Lab7.zip with total 4 files (2 C program files + 2 captured output text files).