CS344: Operating Systems Lab

Lab # 06 (1 Questions, 60 Points)
Held on 03-Oct-2023
Lab Timings: 09:00 to 12:00 Hours Pages: 3
Submission: 12:00 Hrs, 03-Oct-2023
Instructor Dr. V. Vijaya saradhi
Head TAs Adithya Moorthy & Laxita Agrawal
Department of CSE, IIT Guwahati

a. This assignment is based on chapter 3, `Process Management` in the book `Operating System Principles, Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne`.

b. In order to perform this assignment, understanding of system calls `ftok()`, `msgget()`, `msgsnd()`, `msgrcv()`, `msgctl()`, `fork()` are essential.

c. Read the attached resource for a understanding about shared memory

d. Carefully read the manual pages for the above system calls.

Question 1: (60 points)

Interprocess communication - message passing: Implement the following:

Message Passing Given an input file `week06-ML2-input.txt` containing a gray scale image of size 256 × 256, the objective is to perform edge detection. Details to perform edge detection are given in separate PDF file `edge-detection.pdf`. In order to perform this task,

a. (20 marks) 1^{st} C program
   - (5 marks) Obtain a key
   - (5 marks) Create a message queue
   - (5 marks) Construct a message that contain `(row_number, column_number)`. Write the message into the message queue.
   - (5 marks) Remove the message queue

b. (40 marks) 2^{nd} C program
   - (10 marks) Read
     - (2 marks) the size of matrix K_x, read matrix K_x
     - (2 marks) the size of matrix K_y, read matrix K_y
     - (6 marks) the size of image I, read the image I
   - (5 marks) Obtain a key
   - (5 marks) Get access to the message queue of 1^{st} program
   - (10 marks) Perform the following tasks:
     i. Read message from message queue.

ii. If parent has already created EIGHT children, then wait for one of the child to complete the work.

iii. If the parent has less than eight children, then create a new child process

iv. Pass the nine image elements corresponding to the message and nine values corresponding to the kernel matrix. That is:

    i. image/kernel(`(row_number - 1, column_number + 0)`)
    ii. image/kernel(`(row_number + 0, column_number + 0)`)
    iii. image/kernel(`(row_number + 1, column_number + 0)`)

    iv. image/kernel(`(row_number - 1, column_number + 1)`)
    v. image/kernel(`(row_number + 0, column_number + 1)`)
    vi. image/kernel(`(row_number + 1, column_number + 1)`)

    vii. image/kernel(`(row_number - 1, column_number - 1)`)
    viii. image/kernel(`(row_number + 0, column_number - 1)`)
    ix. image/kernel(`(row_number + 1, column_number - 1)`)

v. Compute the edge value at `(row_number, column_number)`

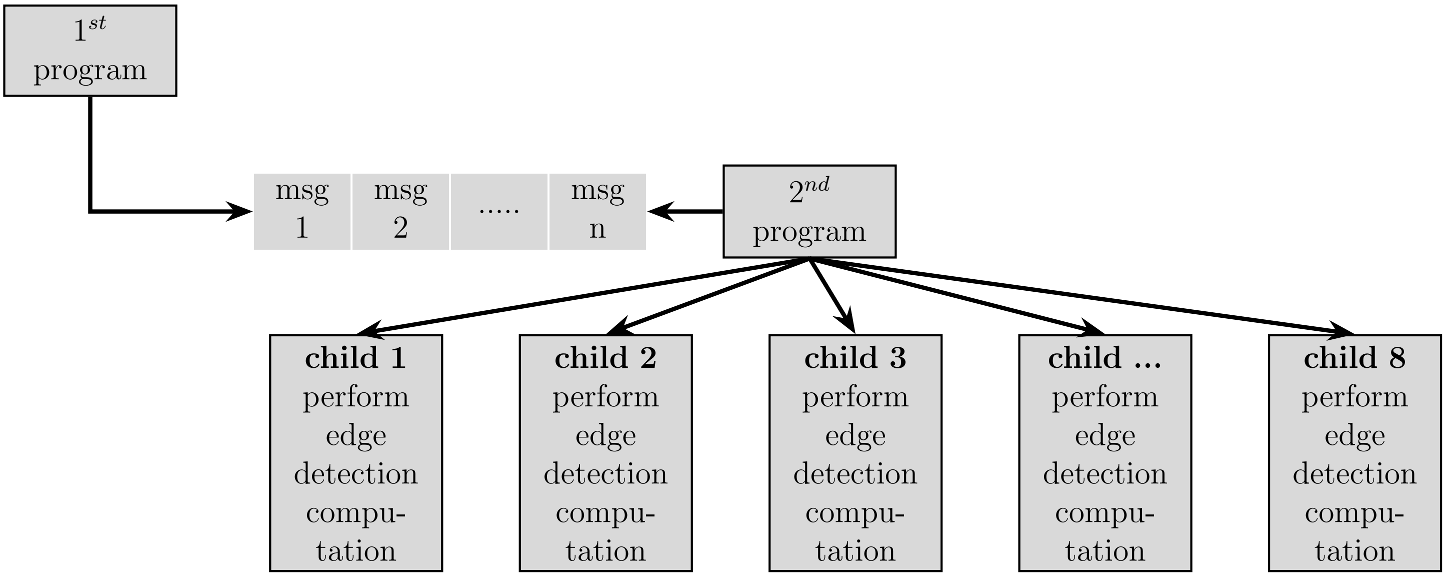
vi. Print the output into a file as: `(row_number, column_number): edge value`

vii. Terminate the child

Ensure that edge value is computed for all the image elements that is:

i. Edge value at (1, 1)
ii. Edge value at (1, 2)
iii. Edge value at (1, 3)
iv. ⋮
v. Edge value at (1, 256)
vi. Edge value at (2, 1)
vii. Edge value at (2, 2)
viii. Edge value at (2, 3)
ix. ⋮
x. Edge value at (2, 256)
xi. ⋮
xii. Edge value at (256, 1)
xiii. Edge value at (256, 2)
xiv. Edge value at (256, 3)
xv. ⋮
xvi. Edge value at (256, 256)

- (10 marks) Sort the output file as per order given above `(i)` to `(xvi)`.


1^{st} program
msg 1
msg 2
.....
msg n
2^{nd} program
child 1 perform edge detection computation
child 2 perform edge detection computation
child 3 perform edge detection computation
child ... perform edge detection computation
child 8 perform edge detection computation