## Tutorial 4

**Question 1:** Explain the circumstances under which the line of code marked printf("LINE J") in Figure 3.22 will be reached.

Only if execlp() failed (e.g: program in parameter not found)

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
       fprintf(stderr, "Fork Failed");
       return 1;
    }
    else if (pid == 0) { /* child process */
       execlp("/bin/ls","ls",NULL);
       printf("LINE J");
    }
    else { /* parent process */
       /* parent will wait for the child to complete */
       wait(NULL);
       printf("Child Complete");
    }

    return 0;
}
```

مو بالضرورة، الأب أسرع.

→ of the child who starts the call.

**Figure 3.22** When will LINE J be reached?

**Question 2:** trace the execution of the program in Figure 3.23, identify the values of pid at lines A, B, C, and D. (Assume that the actual pids of the parent and child are 2600 and 2603, respectively.)

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid, pid1;

   /* fork a child process */
   pid = fork();

   if (pid < 0) { /* error occurred */
      fprintf(stderr, "Fork Failed");
      return 1;
   }
   else if (pid == 0) { /* child process */
      pid1 = getpid();
      printf("child: pid = %d",pid); /* A */
      printf("child: pid1 = %d",pid1); /* B */
   }
   else { /* parent process */
      pid1 = getpid();
      printf("parent: pid = %d",pid); /* C */
      printf("parent: pid1 = %d",pid1); /* D */
      wait(NULL);
   }

   return 0;
}
```

return id of current running process

if child process is working → 2603

The only difference is that the value of the variable pid for the child process is zero, while that for the parent is an integer value greater than zero (in fact, it is the actual pid of the child process).

child: pid = 0
child : pid1 = 2603
parent : pid = 2603
parent : pid1 = 2600

مستحيل يتزامن A مع B
لأنه مستقل . A مع C
بس ممكن
لوين نطبع بالترتيب.

Figure 3.23 What are the pid values?

**Question 3:** Describe the pros and cons with respect to both system level and programmer level for each of the following message-passing alternatives.

a. Synchronous and asynchronous communication

A benefit of synchronous communication is that it allows a rendezvous between the sender and receiver. A disadvantage of a blocking send is that a rendezvous may not be required and the message could be delivered asynchronously.

b. Automatic and explicit buffering

Zero capacity (or explicit )No messages are queued on a link.

? Sender must wait for receiver (rendezvous)

2. Bounded capacity ? Finite length of n messages ? Sender must wait if link full

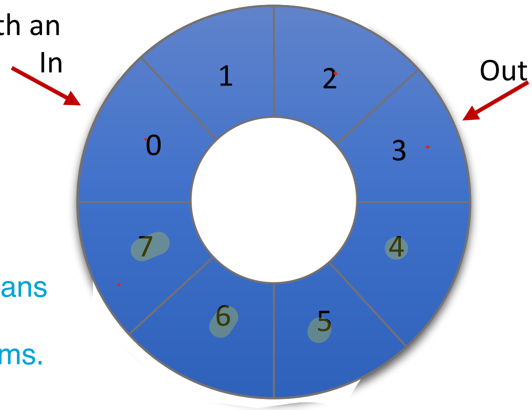
3. Unbounded capacity ? Infinite length ? Sender never waits

c. Fixed-sized and variable-sized messages

fixed-size messages, a buffer with a specific size can hold a known number of messages. The number of variable-sized messages that can be held by such a buffer is unknown. Larger messages (i.e. variable-sized messages) use shared memory to pass the message.

If only fixed-sized messages can be sent, the system-level implementation is straightforward. This restriction, however, makes the task of programming more difficult. Conversely, variable-sized messages require a more complex system-level implementation, but the programming task becomes simpler.

**Question 4:** Given a bounded circular buffer of size =8 with an initial state as shown:




In yellow means they already filled with items.

| SR | Action | In | Out | Full | Empty | Comment |
|---|---|---|---|---|---|---|
| 1 | Initial state | 0 | 3 | No | No | |
| 2 | Produce 2 Means 2 items | 2 | 3 | Yes | No | |
| 3 | Consume 1 | 2 | 4 | No | No | |
| 4 | Produce 2 | 3 | 4 | Yes | No | The buffer is full after the first write. So, the second write is waiting until consuming. |
| 5 | Consume 3 | 4 | 7 | No | No | The previous producer process resumes and writes the item into slot 3 and updates "in" |
| 6 | Consume 5 | 4 | 4 | No | Yes | The buffer is ready read by consumer 5 times. |

Complete the table below that updates the values of both in and out pointers after each action in the table. Note that the Action Produce/Consume x means to perform Produce/Consume x times. Add any assumptions, if necessary.