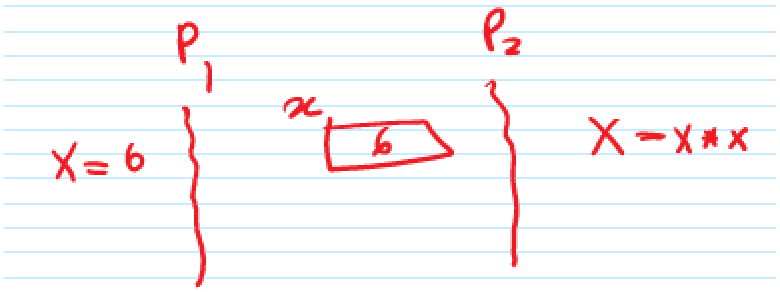
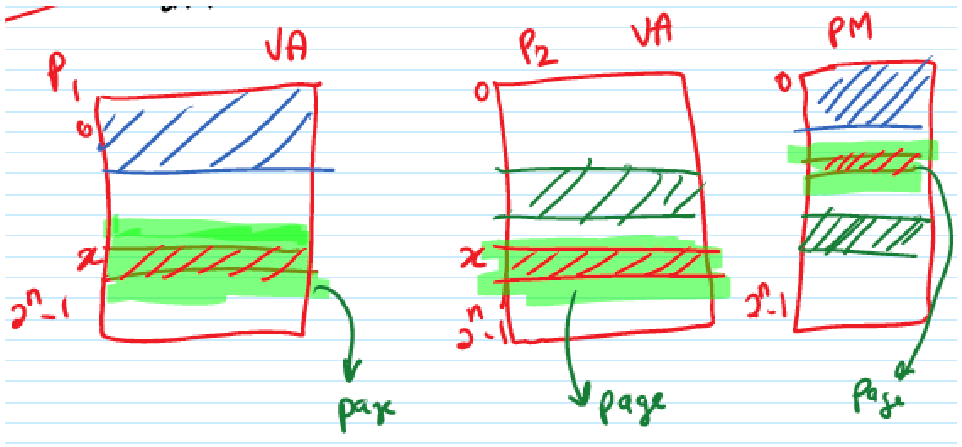
Lecture 7: Concurrent Programming Intro

Inter Process Communication (IPC):

1. Process can communicate using Files.
    - **Example**: communication between a parent and child process.
    - Child inherits file pointer from parent and can use one File for the parent to write into and one for the child to read from.
2. OS supports something called a pipe.
    - Corresponds to 2 file descriptors (int fd [2]).
3. Process could communicate through variables that are shared between them.
    - Shared variable
    - Private variables



- Problem→ address translation to protect one process from another.
- Solution → OS provides the mechanism.



4. Processes could possibly communicate by sending and receiving messages to each other
    - OS provides the mechanism.
    - Program with shared variables:
        - Consider a 2-process program in which both processes increment a shared variable.

```
        int x = 0;

P1                      P2
x++;                    x++;
```

- Question → what is the value of x? → 2
- Problem → x = 1 or 2?

```
X++;  => in instruction set

LDR    R1, [R2]       }  done in both P1 + P2
ADD    R1, R1, #1.    }
STR    [R2], R1
```

- Why can't X be 1?
  - P1 loads X into R1, increments R1.
  - P2 loads X into register before P1 stores a new value into X.
- Need to synchronize processes that are interacting using shared variables.
- Critical section→ part of a program where a shared variable is accessed.

5. Sometimes process don't explicitly share values to cooperate.

   Example. Synchronization. P1 reads (2 matrices), P2 multiples (2 matrices), P3 writes (a matrix). Process 2 should not start work until Process 1 finishes reading. This is called process synchronization. Synchronization primitives: mutex lock, semaphore, barrier.

Critical Sections:

- Must synchronize process so that they access shared variables one at a time one a critical section.
  - Mutual Exclusion
- Mutex Lock→ synchronization primitives
  - Acquire Lock (L)→ 1. Done before a critical section code; 2. Returns a value when safe for processes to enter critical section.
  - Release Lock→ 1. Done after the critical section; 2. Allows another process to acquire lock.

Implementing Lock:

```
int  L = 0;                // 0 => lock is available
                           // 1 => lock is in use.
AcquireLock (L) :
      while (L==1);
        L= 1;
ReleaseLock (L)
       L= 0;
```

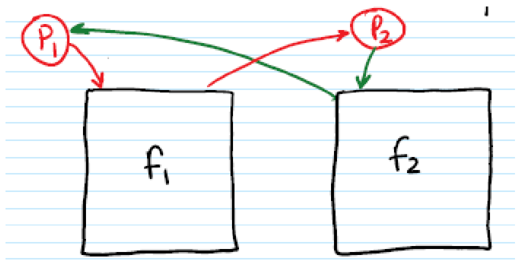
Lecture 8: Concurrent Program Continued

Inter Process Communication (IPC):

- Concurrent programming is about programs with multiple flows of control.
- One way to setup a concurrent program might be to set it up as a program that runs as multiple processes to achieve a common goal.
- To cooperate, process must somehow communicate.

Process Communication:

1. Process can communicate using files.
    a. Example: communication between a parent process and child process.
        i. Parent process creates 2 files before forking child process.
           Why? It's conceivable the parent process may want to be able to send data to the child and the child may want to send the data to the parent.
        ii. Child inherits file descriptors from parent, and they share the file pointers.
        iii. Can use one file for parent to write to and child to read from and the other file for the child to write to and parent to read from.

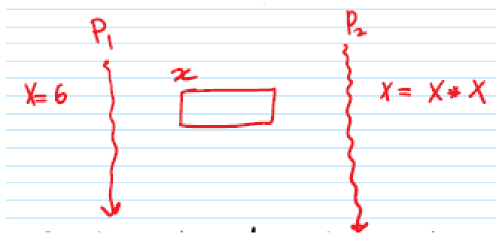


    b. Problem→ Files ~ Hard Disk (both very high overhead)
2. Pipe → technical term
    a. Something supported by all operating systems.
    b. Looks superficially like process can communicate using files.
    c. A pipe provides two file descriptors but not using hard disk.
        int fd [2];

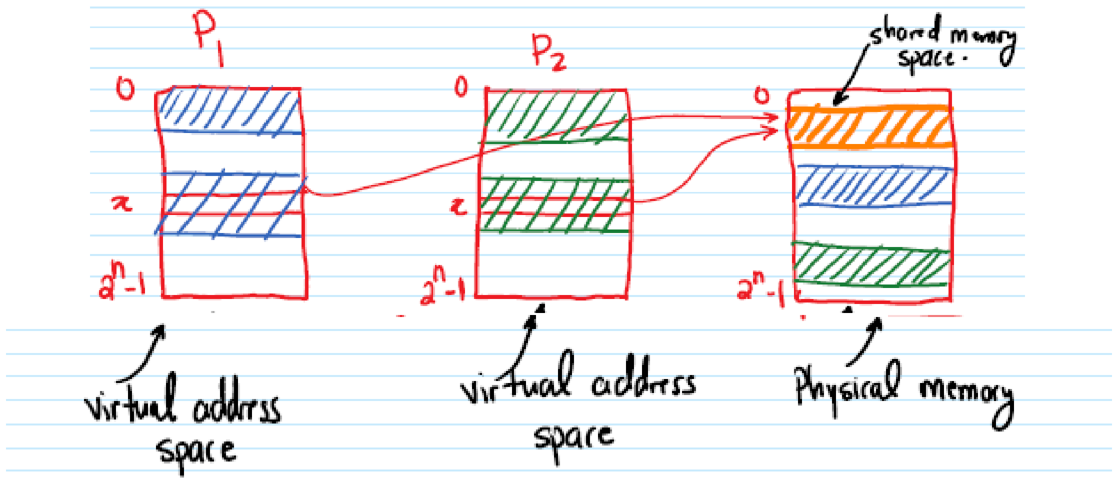
d. Read from fd[0] accesses data written to fd[1] in FIFO (First in First Out) order and vice versa.

3. Processes could communicate through variables that are shared between them

e. There will be some variables that can be used between process P1 and P2 for the purposes of communication and we will refer to them as shared variables and the variables of the process that are not shared, we will refer to them as private variables.



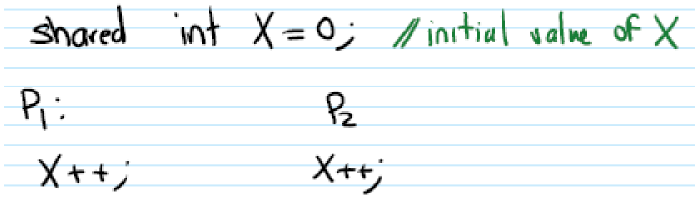
f. Problem: Recall that address translation is used to protect one process from another.



3. Processes could communicate by sending and receiving messages to each other.
    a. Special support for these messages.
4. Sometimes processes may need to communicate but not explicitly communicate values to cooperate.
    a. Processes may just have to synchronize their activities.
    b. Example: Process 1 reads 2 matrices, Process 2 multiplies them, Process 3 write the result to a matrix.
    c. Process 2 should not start work until Process 1 finishes reading.
    d. This is called process synchronization.
    e. To do synchronization, some kind of mechanism has to be provided.
        i. Synchronization primitives:
        Example: mutex lock, semaphores, barrier

Programming with Shared Variables:

- Consider a 2-process program in which both processes increment a shared variable.



```
shared int X = 0; // initial value of X

P1:              P2
 X++;             X++;
```