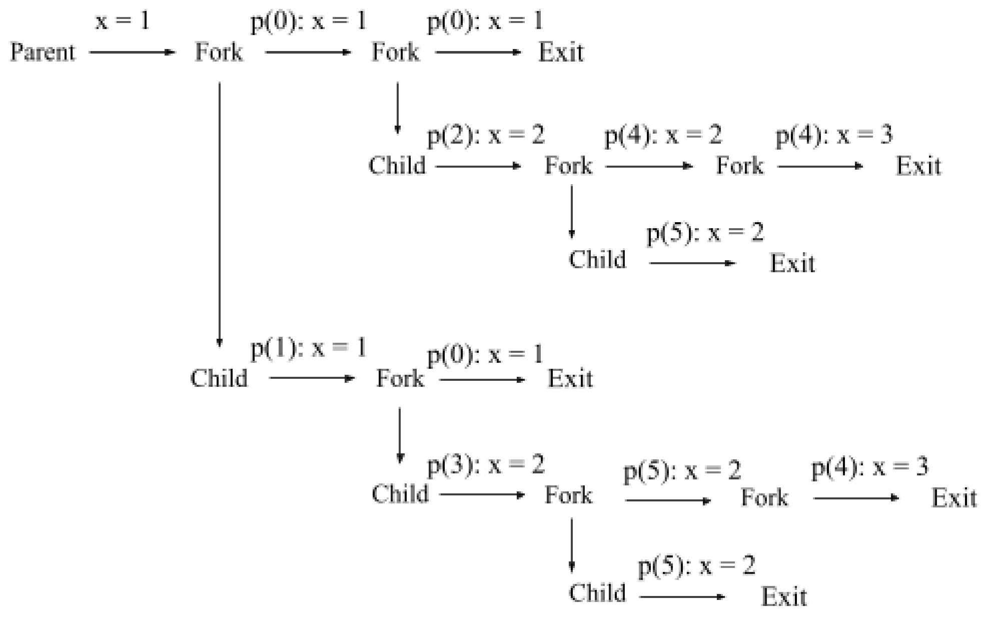
Operating System
Blen Daniel Assefa

OS 2021 Problem Sheet #3

Problem 3.1: process creation using fork()

a)



It creates five processes. On the edges, the values for x are displayed.

Note:

```
if (!fork()) // same as: if (fork() == 0)
{ // child
}else{
 // parent
}
```

b) Produced output:

P0: x = 1
P1: x = 1
P2: x = 3
P4: x = 2
P3: x = 3
P5: x = 2

Problem 3.2: readers / writers problem

a) The situation in which the readers-writers problem fails to work properly is:

Since the last statement of the read function (if (readcount == 0) up(&writer);) is not inside the critical section. This means this statement could maybe be accessed from another reader function that might be simultaneously existing the function. If that situation occurs then the program might face deadlock where that could result in a never signal of the writer function.

b) The situation in which the readers-writers problem fails to work properly is:

The case where if the readers reading the file simultaneously happens to leave the program at the same time. Since the shared variable “writer” is not in the critical section, the program has the option of allowing another reader to access and change the value before it. This might cause the program to be in a deadlock situation.

c) The situation in which the readers-writers problem fails to work properly is:

The case where the reader is reading a file while the writer writes on it, resulting in a deadlock situation. This could happen because since the writer is enclosed to be in a critical section, but the reader is outside the critical section, there could be a window in the function calling where the reader is reading and the writer is also writing.

Problem 3.3: running competition

```
Const int N;
Const int T;
Shared int in = 0;
Semaphore mutex = 1;
Semaphore chips = N; empty_relay = T;
Semaphore full = 0; full_reply = 0;

Shared runner_t buffer[N];
Shared *relay_team team[T];
//Malloc for the list of teams


runners(){
      down(&chips);
      down(&mutex);
      buffer[in++]= runner;
      up(&mutex);
      up(&full);

      // Running code here

      down(&full);
      down(&mutex);
      Buffer[in--] = NULL;
      up(&mutex);
      up(&chips);
}

relay(){

      down(&empty_relay);
      down(&mutex);
      while(*relay_team){
            if(*relay_team != T){
                  *relay_team++;
                  up(&empty_relay);
            }else{
                  up(&full_relay);
            }
      }
      up(&mutex);

      // This is for getting the chip for the team

      down(&full_relay);
      down(&mutex);
      // get chips for T players
      up(&mutex);
      up(&empty_relay);

}
```