OS HW1 Example

Include

```
#include <thread>
#include <vector>
#include <stdio.h>
#include <string.h>        // memset
#include <unistd.h>        // getpid, gettid, fork
#include <sys/ipc.h>       // ftok
#include <sys/shm.h>       // shm
#include <sys/syscall.h>   // syscall
```

- #define gettid() syscall(SYS_gettid)

- int gRESULTS_T[1000];
- int gRESULTS_P[1000];

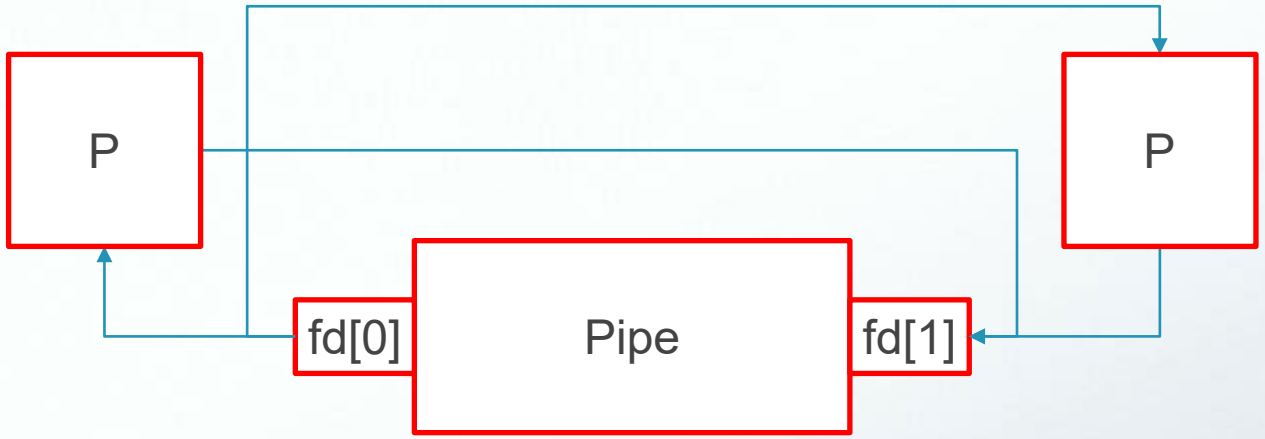
Multi Thread

```
vector <thread> threads;
for (int i = 1; i <= num; i++)  threads.push_back(thread(FibonacciT, i, i));
for (int i = 1; i <= num; i++) threads[i-1].join();
for ( int i = 0; i < num ; i++ )  printf( " %d,", gRESULTS_T[i] );
```

Multi Process – PIPE

```
pid_t pid;
int keep = 0;
int fds[num][2];
for (int i = 1; i <= num ; i++) {
  if ( pipe(fds[i-1]) < 0 ) printf( "Pipe error" );
  pid = fork();
  keep = i;
  if (pid == 0 || pid == -1) break;
}
```

Pipe



- Write:
- close(fd[0]);
- write(fd[1], &result, sizeof(int));
- Read:
- close(fds[1]);
- read(fds[0], &gRESULTS_P[i], sizeof(int));

Multiprocess

```
if ( pid == 0 ){
  FibonacciP(fds[keep-1], keep, keep);
  exit(0);
}
```

```
else{
 for ( int i = 1; i <= num; i++ ) pid =  wait(NULL);
  for ( int i = 0; i < num ; i++ ) {
      close( fds[i][1] );
      read( fds[i][0], &gRESULTS_P[i], sizeof(int) );
  }
  for ( int i = 0; i < num ; i++ ) {
      printf( " %d,", gRESULTS_P[i] );
  }
}
```

Multiprocess-shared memory

```
+ pid_t pid;
+ int keep = 0;
+ key_t key = ftok("main.cpp",16);
+ int shmid = shmget(key,1000,0666|IPC_CREAT);
+ int arr_a[1000];
+ int *arr_b = (int*) shmat(shmid,(void*)0,0);
```

From: https://www.geeksforgeeks.org/ipc-shared-memory/

Multiprocess-shared memory

```
+    for (int i = 1; i <= num ; i++) {
+      pid = fork();
+      keep = i;
+      if (pid == 0 || pid == -1) break;
+    }
```

Multiprocess-shared memory

```
+ if ( pid == 0 ){
+   FibonacciP_shm(arr_a, arr_b, num, keep, keep);
+   shmdt(arr_b);
+   exit(0);
+ }
```

Multiprocess-shared memory

```
+ else{
+   for ( int i = 1; i <= num; i++ ) pid =  wait(NULL);
+   for ( int i = 0; i < num ; i++ ) printf( " %d,", arr_a[i] );
+   for ( int i = 0; i < num ; i++ ) printf( " %d,", arr_b[i] );
+   shmdt(arr_b);
+   shmctl(shmid,IPC_RMID,NULL);
+ }
```

Compile

- g++ -pthread main.cpp
- pthread：link library

Multi thread

```
root@server1:/home/cdlab/OS-HW1-EXAMPLE-v1# ./a.out
Input numbers in the range of ( 1, 10 ):
10
[24220, 24223] From 2th threads Fibonacci is: 1
[24220, 24222] From 1th threads Fibonacci is: 1
[24220, 24224] From 3th threads Fibonacci is: 2
[24220, 24225] From 4th threads Fibonacci is: 3
[24220, 24226] From 5th threads Fibonacci is: 5
[24220, 24227] From 6th threads Fibonacci is: 8
[24220, 24228] From 7th threads Fibonacci is: 13
[24220, 24229] From 8th threads Fibonacci is: 21
[24220, 24230] From 9th threads Fibonacci is: 34
[24220, 24231] From 10th threads Fibonacci is: 55
Result: [ 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 ]
==================================================
==================================================
[24232, 24232] From 1th process Fibonacci is: 1
[24233, 24233] From 2th process Fibonacci is: 1
[24234, 24234] From 3th process Fibonacci is: 2
[24235, 24235] From 4th process Fibonacci is: 3
[24236, 24236] From 5th process Fibonacci is: 5
[24237, 24237] From 6th process Fibonacci is: 8
[24238, 24238] From 7th process Fibonacci is: 13
[24239, 24239] From 8th process Fibonacci is: 21
[24240, 24240] From 9th process Fibonacci is: 34
[24241, 24241] From 10th process Fibonacci is: 55
Result: [ 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 ]
```

gRESULTS_T

[pid , tid]

Multi process

gRESULTS_P