OS Project1

- 姓名 : 林哲宇
- 學號 : 0616018

Task2

題目要求建造一個筆直的十層的樹，並且要求要在 child 結束後，parent 才可以結束。

```
int main(){
  int status;
  int mainp = getpid();
  pid_t pid = 0;
  for(int i=0; i<11; i++){
    if(pid < 0){
      printf("fail");
    }
    else if(pid == 0){
      printf("process pid %d create\n", getpid());
      pid = fork();
    }
    else{
      wait(&status);
      printf("process pid %d exit\n", getpid());
      exit(0);
    }
  }
  wait(&status);
  if(pid)printf("process pid %d exit\n", getpid());
}
```

於是我用迴圈跑十一次，因為第一次是 main process。當執行完 `process pid XXX create` 之後就會 fork 出一個 child。

這時 parent process 的 pid 會大於 0，而 child process 的 pid 會等於 0。當 parent process 繼續跑迴圈第二次時，就會跑到 else，並且等待 child 跑完。直到 child 跑完，才執行 `process pid XXX exit` ，接著就 exit(0)。

然而如果只有做到這樣，最下面的 child 不會印出 `process pid XXX exit` 。所以最後面要再補一次，結果才會如 spec 要求的。

Task3

這題連續 fork 三個 process。第一個 x 被 fork 出來之後，又會繼續 fork y, z。跟上一題一樣要求我們要在 child 結束後，parent 才能結束。

```
int main(int argc ,char* argv[]){
    int pidx=0,pidy=0,pidz=0;
    printf("create main process %d\n",getpid());
    printf("process %d create process %d\n",getpid(),pidx=fork());
    printf("process %d create process %d\n",getpid(),pidy=fork());
    printf("process %d create process %d\n",getpid(),pidz=fork());
    /* -------------- */
    //printf("getpid %d => x : %d, y : %d, z : %d\n", getpid(), pidx, pidy, pidz);
    int x = pidx, y = pidy, z = pidz;
    if(pidy == 0) x = 0;
    else if(pidz ==0 ){
      x = 0;
      y = 0;
    }
    wait(&pidx);
    wait(&pidy);
    wait(&pidz);
    pidx = x, pidy = y, pidz = z;
    /* -------------- */
    printf("process %d exit its child process %d %d %d\n",getpid(),pidx,pidy,pidz);
}
```

在兩行多行註解中間的部分是我自己打的，其餘是作業提供的。

應 spec 要求， pidz 被創出來後，前面的 pidx, pidy 必須是 0。所以我設了一些條件判斷，讓被 fork 出來的 pid 之前的 pid 值為 0。

接著就是直接 wait pidx, pidy, pidz，結果就會是等到 child 結束，parent 才結束了。

Task5

題目給了一個 judge 和 sol 的範本與用來測試的 interactive.sh (http://interactive.sh)。要求用我們的 sol 和 TA 的 judge 進行猜拳。最後分數就是這題的分數。

```
int main (int argc , char **argv){
    int shmid = 0;
    int *shm;
    try{
        key_t key_value =  ftok("/tmp/123.txt", 123);
        cout << key_value << endl;
        shmid = shmget(key_value, sizeof(int), IPC_CREAT|0666);
        shm = (int*)shmat(shmid, NULL, 0);
        const string s = "Paper";
        int ti = 100;
        int score = 0;
        while(ti--){
            *shm = 0;
            cout << "OK" <<endl;
            string opponent_option ;
            cin >> opponent_option;
            cout << s <<endl;
            if(opponent_option == "Scissor")
                score++;
        }
        cerr << score << endl;
        shmdt(shm);
        shmctl(shmid,IPC_RMID,0);
    }
    catch(...){
        cerr << 0 <<endl;
        if(shm != 0){
            shmdt(shm);
            shmctl(shmid,IPC_RMID,0);
        }
    }
}
```

這是 TA 的 fake judge，就是建立一個 shared memory，並把 key 印出來，之後印個 OK 之後就猜拳，贏了的話分數就加一，執行一百次。spec 有說會把要出的拳在我們出拳前放到 shared memory。

```
string fist[3]={"Paper","Scissor","Stone"};
int main (int argc , char **argv){
    int shmid = 0;
    int *shm;
    try{
        key_t key_value =  ftok("/tmp/123.txt", 123);
        cout << key_value << endl;
        shmid = shmget(key_value, sizeof(int), IPC_CREAT|0666);
        shm = (int*)shmat(shmid, NULL, 0);
        int ti = 100;
        int score = 0;
        while(ti--){
            *shm = rand()%3;
            cout << "OK" <<endl;
            string opponent_option ;
            cin >> opponent_option;
            cout << fist[*shm] <<endl;
            if(opponent_option == fist[(*shm+1)%3])score++;
        }
        cerr << score << endl;
        shmdt(shm);
        shmctl(shmid,IPC_RMID,0);
    }
    catch(...){
        cerr << 0 <<endl;
        if(shm != 0){
            shmdt(shm);
            shmctl(shmid,IPC_RMID,0);
        }
    }
}
```

我把 judge 改一改，把它變成在出拳之前，先把隨機的 0~3 存入 shared memory，之後依照結果來決定是否加分。

```
string fist[3] = {"Paper", "Scissor", "Stone"};

int main(){
  int *shmaddr, key, shmid;
  string OK;
  cin >> key;
  shmid = shmget(key, sizeof(key_t), 0);
  shmaddr = (int *) shmat(shmid, NULL, 0);
  for(int i=0; i<100; i++){
    cin >> OK;
    cout << fist[(*shmaddr + 1) % 3] << endl;
    cin >> OK;
  }
}
```

這是我寫的 sol。就是每次出拳之前把 shared memory 的東西讀進來之後，把贏它的拳 cout。並且因為 judge 會印一些東西，所以我用字串變數 OK 去接收。

編譯之後，./interactive.sh ./judge ./task5，印出 100。

心得

task2, task3 讓我學到 parent 和 child 的關係，利用 c++ fork 出其他 process，並且藉由觀察它們讓我對 process 有更多的理解。

task5 讓我學到 shared memory 的知識，利用共同存取空間，可以讓兩個 process 進行溝通。

這次作業不難，但是我覺得 spec 很多事情可能講得沒有很清楚，要去猜題意這部分花了比較多時間。