

操作系统实验报告

顾芄骐 20079100001

2023 年 4 月 10 日

1 进程的建立

- 实验目的：学会通过基本的 Windows 或者 Linux 进程控制函数，由父进程创建子进程，并实现父子进程协同工作。
- 实验软件环境：Ubuntu22.04 & GNU-C++17
- 实验内容：创建两个进程，让子进程读取一个文件，父进程等待子进程读取完文件后继续执行，实现进程协同工作。进程协同工作就是协调好两个进程，使之安排好先后次序并以此执行，可以用等待函数来实现这一点。当需要等待子进程运行结束时，可在父进程中调用等待函数。

1.1 代码实现

在“*os1.cpp*”中写如下代码，并创建名为“*os1.in*”的文件，在后者中写入且仅写入“TextMessage”作为测试输出内容。

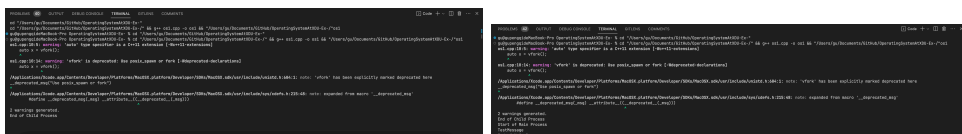
```

1 #include<fstream>
2 #include<unistd.h>
3 #include<stdio.h>
4 #include<iostream>
5 using namespace std;
6 int main(){
7     int flag = 0;
8     string answer = "";
9     auto x = vfork();
10    if (!x) {
11        ifstream in("os1.in");
12        in >> answer;
13        in.close();
14        flag = 1;
15        cout << "End of Child Process" << endl;
16    } else {
17        while (!flag);
18        cout << "Start of Main Process" << endl;
19        cout << answer << endl;
20        cout << "End of Main Process" << endl;
21    }
22    return EXIT_SUCCESS;
23 }

```

1.2 实验结果

运行上述代码，可得到如下左图所示的实验结果。片刻之后，终端显示如右所示的结果。



1.3 实验结果分析

在 1.1 所示代码中，子进程读取了文件“*os1.in*”；读取完成后，父进程继续执行，并显示了开始、结束和子进程所读取的文件内容。需要注意的是，使用 `vfork` 时修改静态区变量是未定义行为，会引发“栈溢出（stack smashing）”。

2 线程共享进程数据

- 实验目的：了解线程与进程之间的数据共享关系。创建一个线程，在线程中更改进程中的数据。
- 实验软件环境: Ubuntu22.04 & gnu-c++17
- 实验内容: 在进程中定义全局共享数据，在线程中直接引用该数据进行更改并输出该数据。

2.1 代码实现

在“*os2.cpp*”中写如下代码。

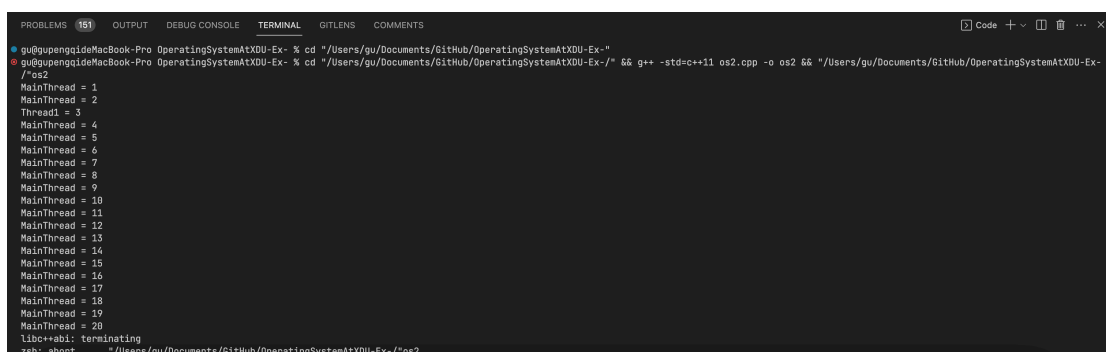
```
1 #include <thread>
2 #include <mutex>
3 #include <bits/stdc++.h>
4 #include <unistd.h>
5 using namespace std;
6 static int Sample = 1;
7 mutex mtx;
8 void thread1(int n){
9     while (Sample <= n)
10     {
11         if (mtx.try_lock())
12         {
13             cout << "Thread1 " << Sample << "\n";
14             sleep(1);
15             Sample++;
16             mtx.unlock();
17         }
18     }
19 }
20 int main(){
21     int n = 20;
22     thread t1(thread1, n);
23     while (Sample <= n){
24         if (mtx.try_lock()){
25             cout << "MainThread " << Sample << "\n";
26             Sample++;
27             mtx.unlock();
28         }
29     }
30     return 0;
31 }
```

2.2 实验结果

运行上述代码，可得到实验结果如下图所示。

2.3 实验结果分析

在此次实验结果中，当 `Sample` 值为 3 时，执行了 `thread1` 函数。线程中直接引用了这一数据进行更改并输出之。



```
PROBLEMS 151 OUTPUT DEBUG CONSOLE TERMINAL GITLENS COMMENTS
gu@guopengdaMacBook-Pro OperatingSystemAtXDU-Ex- % cd "/Users/gu/Documents/GitHub/OperatingSystemAtXDU-Ex-"
gu@guopengdaMacBook-Pro OperatingSystemAtXDU-Ex- % cd "/Users/gu/Documents/GitHub/OperatingSystemAtXDU-Ex-/"
gu@guopengdaMacBook-Pro OperatingSystemAtXDU-Ex- % g++ -std=c++11 os2.cpp -o os2
MainThread = 1
MainThread = 2
Thread1 = 3
MainThread = 4
MainThread = 5
MainThread = 6
MainThread = 7
MainThread = 8
MainThread = 9
MainThread = 10
MainThread = 11
MainThread = 12
MainThread = 13
MainThread = 14
MainThread = 15
MainThread = 16
MainThread = 17
MainThread = 18
MainThread = 19
MainThread = 20
libc++abi: terminating
^ah: abort "/Users/gu/Documents/GitHub/OperatingSystemAtXDU-Ex-/"os2
```

2.4 反思

本次实验代码中运行结果可能与“实验结果”章节中的不一致，有时会出现 `thread1` 函数不执行的情况，20 个输出结果均为“MainThread=x”。