操作系统作业 2

傅申 PB20000051

1

有 15 个进程会被创建
情况如下, 其中 `child` 后接的是被 `fork()` 时 `i` 的值

```
parent─┬─child0─┬─child1─┬─child2───child3
       │        │        └─child3
       │        ├─child2───child3
       │        └─child3
       ├─child1─┬─child2───child3
       │        └─child3
       ├─child2───child3
       └─child3
```

2

当 `execlp()` 出现错误时, 子进程的内容不会被破坏, 即可达到 `printf("LINE J)`, 在这里可能有以下几种情况会导致这一结果

- `/bin/ls` 文件无法被找到
- `/bin/ls` 无法被访问
- 没有足够的内存去执行 `/bin/ls`

3

A: 0, B: 2603, C: 2603, D:2600

4

输出如下

```
CHILD: 0
CHILD: -1
CHILD: -4
CHILD: -9
CHILD: -16
PARENT: 0
PARENT: 1
PARENT: 2
```

```
PARENT: 3
PARENT: 4
```

因为父进程执行了 `wait(NULL)`, 所以行 Y 的代码会在子进程结束后执行.
在 `fork()` 后, 在用户空间, 父进程的数据 (包括 `nums` 数组) 会复制一份到子进程中, 所以子进程对 `nums` 的修改不会影响到父进程, 而子进程在行 X 之前执行了 `nums[i] *= -i;` 所以在行 X 每次都输出 $-i^2$. 而之后父进程执行, 因为没有受到子进程影响, 在行 Y 仍然输出 i.

5

一般情况下, 行 X 的代码不会被执行, 因为 `execl()` 会加载二进制文件 `/bin/ls` 到内存中, 破坏原本的进程内存内容, 在 `ls` 结束运行后, 进程也就结束了, 所以下一行 (行 X) 的代码不会被执行, 除非出现题 2 中的情况.

6

在进程终止后, 进程并没有被真正移除, 这时进程处于终止状态, 用户空间的内存被释放, 但是在内核空间中仍保存着其在进程表中的条目, 等待父进程 `wait()` 之后, 进程才被真正移除. 所以进程需要一个终止状态.

7

僵尸进程是已经终止但父进程仍未调用 wait() 的进程, 父进程调用 `wait()` 后僵尸进程就会被消灭.

8

- 用户空间
  - 进程内存
  - 程序代码
- 内核空间
  - 内核数据结构
  - 内核代码
  - 设备驱动

9

`exec*()` 系统调用会加载对应的二进制文件到内存中, 破坏原有的进程内存内容, 并开始执行. 而一般的函数调用并不会破坏内存.

10

- 响应性: 增加响应速度
- 资源共享: 线程之间能共享资源

- 经济
- 可伸缩性: 多线程可以在多个处理器核上并行执行

堆和全局变量会被共享, 栈和寄存器值不会.

11

不包括父进程, 有 4 个不同的进程会被创建.

```
parent─┬─child0─┬─child00───child000
       │        └─child01
       │
       └─child1
```

其中 `child0`, `child00` 会创建线程, 所以有 2 个线程会被创建.

12

输出为

```
CHILD: value = 5
PARENT: value = 0
```

子进程输出 5 是因为 `runner()` 线程将全局变量 (线程之间共享) `value` 的值设置为 5.
父进程输出 0 是因为进程之间 `value` 并没有被共享.

13

- 普通管道
  - 需要具有父子关系 (父子进程之间通信)
  - 单向通信
  - 通信结束后即消失
- 命名管道
  - 父子关系不是必须的
  - 可被多个进程使用 (可能有多个写进程)
  - 双向通信 (UNIX 半双工, Windows 全双工)
  - 需要被显式删除