12周作业

17375318 黎可为

1

- 进程是描述程序执行过程和资源共享的基本单位。
- 当我们要执行一个应用程序的时候，我们就会创建一个进程用来执行程序。系统可以为进程分配资源，每个进程之间相互独立，具有独立的地址空间。同时，进程之间也可以进行交互和通信。
- 进程对于Linux来说可以认为是把一个程序实例化，系统有了一个可以管理执行程序的单位。对于接下来的一些程序的执行管理，都是基于进程进行的。系统可以管理进程的生命周期，给进程分配或回收资源，实现多个应用程序之间的协调工作。

2

- 代码段A调用了31次 `fork()` ，最后一共有32个进程；代码段B调用了5次 `fork()` ，最后一共有6个进程。

1. 对于代码段A，在第一次循环时调用 `fork()` 产生了一个子进程；在第二次循环时父进程和子进程分别有调用了一次 `fork()` ，一共产生了2个子进程；在第三次循环的时候，有4个进程调用了 `fork()` ；第四次、第五次循环分别有8个、16个进程调用了 `fork()` ，所以一共是调用了31次 `fork()` 。程序执行到最后加上最初的进程一共有32个进程。
2. 对于代码段B，由于产生的子进程有 `(pid=fork())==0` 为 `1` ，所以子进程会进入 `if` 下面的语句然后跳出循环。于是所有的子进程一旦创建之后就会立刻跳出循环。而父类的 `pid` 的值为子进程的 `pid` ，不为0。因此调用了 `fork()` 的进程就只有父进程，所以 `fork()` 只被调用了5次。程序执行到最后加上最初的进程一共有6个进程。

3

- 当一个进程终止后，如果没被正确的清除，依然占用着系统的资源，那么这个进程就是一个僵尸进程。
- 当一个进程存在着子进程，则在执行了 `wait()` 之后会进入休眠状态等待子进程终止。当任意一个子进程终止时，父进程会立刻被唤醒， `wait()` 函数返回子进程的 `pid` ，并把子进程的退出状态存放在 `wait()` 函数的参数中。之后继续执行接下来的代码。
- 如果父进程在调用了 `wait()` 前已经有子进程终止了，那么在调用 `wait()` 之后函数会立刻返回其中一个已经终止的子进程的 `pid` ，父进程会立刻被唤醒。

4

- 信号作为一种进程通讯机制，可以让进程之间相互发送或接收一个信号值。
- 利用信号机制，可以控制一些进程的行为。比如说终止、强制终止等。它在软件层面上模拟了硬件层面上的中断机制。
- 利用信号机制，还可以对异常情况进行一些处理。当进程接收到信号时，可以立即进行处理，并运行相关的信号处理函数。

5

- 当一个信号产生后，却没有传递给任何进程，此时信号就处于未决状态。
- 信号处于未决状态通常是由于接收的进程对信号进行阻塞导致的。
- 类似与中断屏蔽，当一个进程在执行某一段代码时不想被信号干扰时，可以对信号进行阻塞。此时，系统会给被阻塞的信号保持为未决状态。而当进程取消阻塞信号后，可以对此信号进行处理。当然也可以忽略这个信号。未决状态的机制可以让进程在一段时间内屏蔽所有的信号，并且在之后还可以重新接收并处理被屏蔽的信号。

6

1. 设置一个信号量 `sem_mutex` 。
2. 当一个进程要进行读写操作时，可以对 `sem_mutex` 进行 `P` 操作，表示占用资源。这样可以防止其他的进程对其进行读写，同时也保证了在其它进程进行读写时会等待读写结束后在进行写操作。
3. 读写结束后，可以对 `sem_mutex` 进行 `V` 操作，表示释放（取消占用）资源，使得其他进程可以继续对共享内存进行读写操作。