OS_lyk_lab03 问答题

written by 涩狗

> 说明:
>
> 1. 作者对使用或不使用本文档造成的问题不负任何责任, 不对文档正确性做出任何担保.
> 2. 答案来源是实验文档, 上网查阅资料, 还有同学交流.
> 3. 希望大家不要死记硬背, 执行并理解后再进行记忆.

1. 解释 `wc` 和 `grep` 指令的含义.

- wc: 计算文件的字数/列数等
- grep: 查找文件中符合条件的字符串

2. 解释 `ps aux | grep firefox | wc -l` 的含义。

显示当前运行的所有进程名带 `firefox` 的进程数量.

3. `echo aaa | echo bbb | echo ccc` 是否适合做shell实验中管道符的检查用例？说明原因。

不适合.

因为 `echo` 命令本身不接受输入, 所以前两个 `echo` 的结果不会显示, 最后的结果和单独执行 `echo ccc` 是一样的, 无法说明多管道的正确性.

4. 对于匿名管道，如果写端不关闭，并且不写，读端会怎样?

- 读端读完之后会继续等待,此时阻塞,直到有数据写入才继续
- 尤其地, 假如一条管道有多个写端, 那么只有在所有写端都关闭之后(管道的引用数降为0), 读端才会解除阻塞状态

5. 对于匿名管道，如果读端关闭，但写端仍尝试写入，写端会怎样?

写进程收到信号 `SIGPIPE`, 通常导致进程异常中止.

6. 假如使用匿名管道从父进程向子进程传输数据，这时子进程不写数据，为什么子进程要关闭管道的写端?

为了防止父进程数据传输完毕(父进程写端关闭)之后, 若子进程写端仍打开, 可能出现的阻塞现象.

7. `fork` 之后, 是管道从一分为二, 变成两根管道了吗? 如果不是, 复制的是什么?

不是.

`fork` 函数作用是创建子进程, 子进程继承了与父进程相同的结构体/代码等, 因此从管道角度实际上是复制了父进程的 `pipefd` 数组(包含 `fd[0]` 和 `fd[1]`), 用的还是一根管道.

8. 解释系统调用 `dup2` 的作用

该函数相当于将 `newfd` 标识符变成 `oldfd` 的一个拷贝，与 `newfd` 相关的输入/输出都会重定向到 `oldfd` 中.

9. 什么是shell内置指令, 为什么不能 `fork` 一个子进程然后 `exec cd`?

- 就是由 Bash 自身提供的命令，而不是文件系统中的某个可执行文件.
- 因为子进程执行结束后会回到了父shell环境, 而父shell的路径根本没有被改变, 最终无法得到期望的结果.

10. 为什么 `ps aux | wc -l` 得出的结果比 `get_ps_num` 多2?

- `ps aux`的表头会占一行
- `ps aux`本身也是一个进程, 运行命令`ps aux`时会输出它的相关信息, 也占一行, 因此一共多了两行

11. 进程名的最大长度是多少? 这个长度在哪定义?

最大长度为`16`.定义在`sched.h`中.

源代码为`#define TASK_COMM_LEN 16`

12.`task_struct`在Linux源码的哪个文件中定义?

`sched.h`

13. 为什么无法通过`SYSCALL_DEFINEx`定义二维数组（如`char (*p)[50]`）为参数?

已被删除.(其实我也不知道为什么, 群里助教给出了答案)

14. 在修改内核代码的时候, 能用 printf 调试吗? 如果不能, 应该用什么调试?

不能. `printf`应当在用户态使用, 内核态应该使用`printk`.

15.`read()`、`write()`、`dup2()`都能直接调用。现在我们已经写好了一个名为`ps_counter`的系统调用。为什么我们 不能在测试代码中直接调`ps_counter()`来调用系统调用?

`read`等系统调用已经被封装在`glibc`库中了, 但是`ps_counter`还没有, 因此不能直接调用, 而是要通过调用`glibc`中的`syscall`库函数来使用.

- `glibc`是GNU发布的`libc`库，即C运行库。`glibc`是linux系统中最底层的api，几乎其它任何运行库都会依赖于`glibc`。`glibc`除了封装linux操作系统所提供的系统服务外，它本身也提供了许多其它一些必要功能服务的实现。