操作系统实验 1

|  |  |  |
| --- | --- | --- |
| 姓名：崔茜宁 | 学号：2019011571 | 班级：计算机19-4班 |

# Unix外壳和历史特征

## 创建子进程

在这个项目中，父进程负责读取用户输入的命令并处理成合法的格式，然后创建子进程来执行命令。

为了让程序的结构更清晰，定义读取命令、处理命令、执行命令等函数，下面依次介绍它们的实现方法和功能。

int get\_command(char command\_buffer[], char \*args[], int \*background);

读取命令函数，由父进程提前执行，command\_buffer 用于保存用户输入的命令，args 为 execvp() 函数的参数，它的值通过后续使用处理命令来获得， background 用来标记命令末尾是否有 & 符号。

该函数使用 read() 从标准流读取一个限制最大长度的字符串，保存在 command\_buffer 中，紧接着函数将 command\_buffer 传递给处理命令函数。

int process\_command(char command\_buffer[], char \*args[], int \*background);

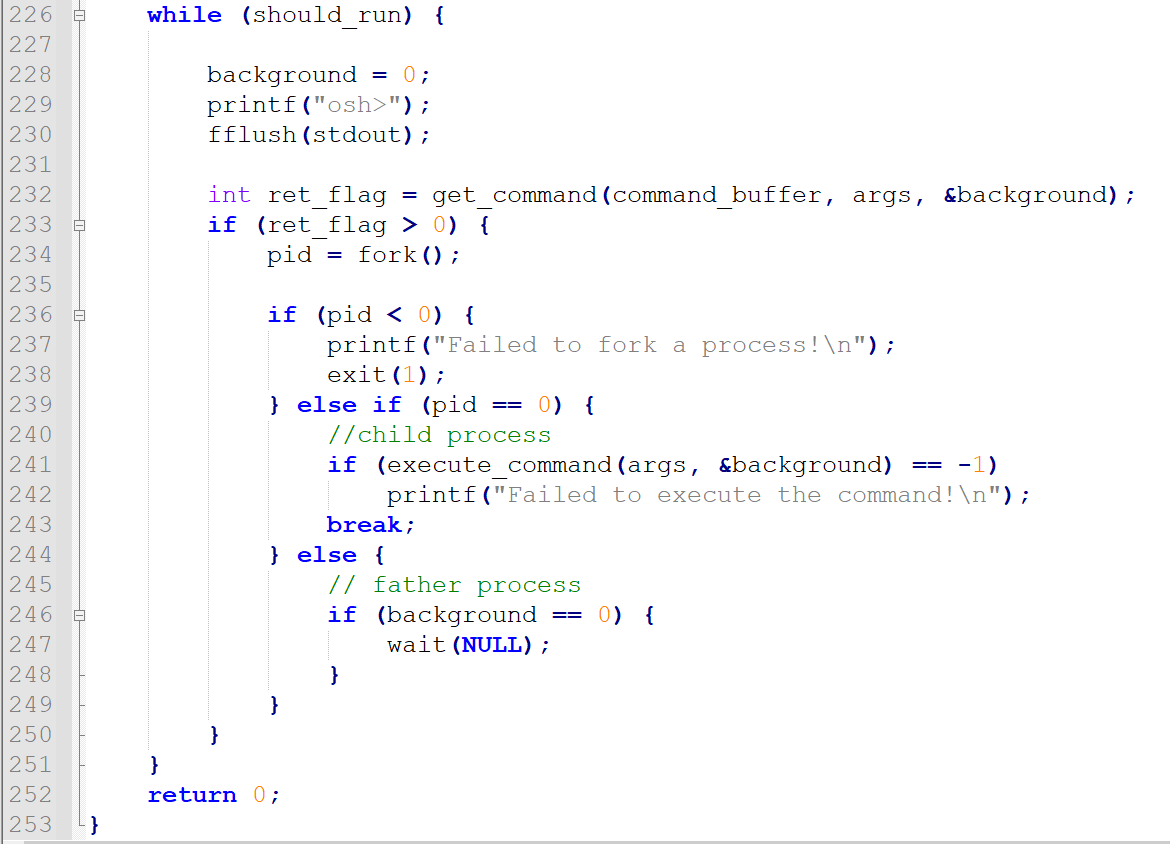
处理命令函数，主要对用户输入的原始命令进行一些处理，如分割、替换等，并将处理好的命令保存在 args 中，后续传递给子进程。

另外如果处理过程发现用户使用了历史命令的功能，即输入 !! 或者 !N 格式的命令，函数还会把相应的历史命令替换过来，然后处理新的命令。

这个函数还包含把命令保存在历史命令列表里的功能，在下一部分介绍。

int execute\_command(char \*args[], int \*background);

执行命令函数，由子进程执行，需要特殊判断命令是否为 history，如果是则调用显示历史命令函数，在下一个部分介绍；否则调用 execvp 函数。



## 创建历史功能

这一部分要完成历史功能，允许用户访问最近输入的命令。这里要求实现的是访问 10 个历史命令，每个命令的编号递增。

数据结构使用的是**循环链表**，增加新的命令比较方便，当历史命令超过 10 条时也能做到覆盖最远的命令。具体的实现如下。

定义链表头和尾的索引 head 与 tail，令 head 始终指向链表中第一个历史命令，即最久使用的那个命令，特殊情况是链表为空时 head 指向 0； 令 tail 始终指向链表中最后一个命令的下一个位置，初始时为 0；还需要定义 cmd\_cnt，表示所有保存过的命令条数。

当需要插入新的历史命令时，可以在 tail 的位置直接插入，然后更新 tail、cmd\_cnt；另外如果插入的位置在 head 处，说明未插入时链表就满了，那么新的命令就覆盖了最久使用的命令，还要更新 head。

显示历史命令的功能通过 void display\_commands\_history() 函数实现，该函数从链表尾部向首部的方向遍历，显示在屏幕上即最近使用的命令显示在上方。

下图展示了一些命令的效果。

1 处为编写 makefile 文件，使用 make 命令进行编译；

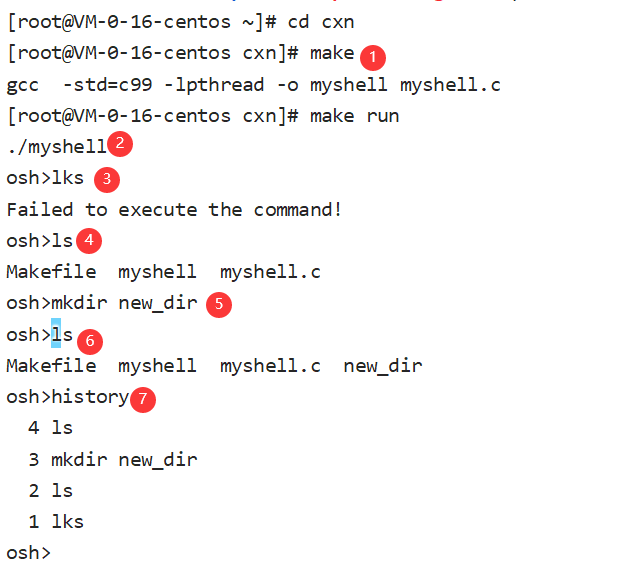
2 处为执行程序；

3 处是错误命令，有相应的错误信息；

4 处是 ls 命令，下一行显示当前目录的信息；

5 处为 mkdir 创建文件夹命令， 6 处可以看到有新增的文件夹；

7 处为历史命令，可以看到历史命令列表不会保存 history 命令本身。



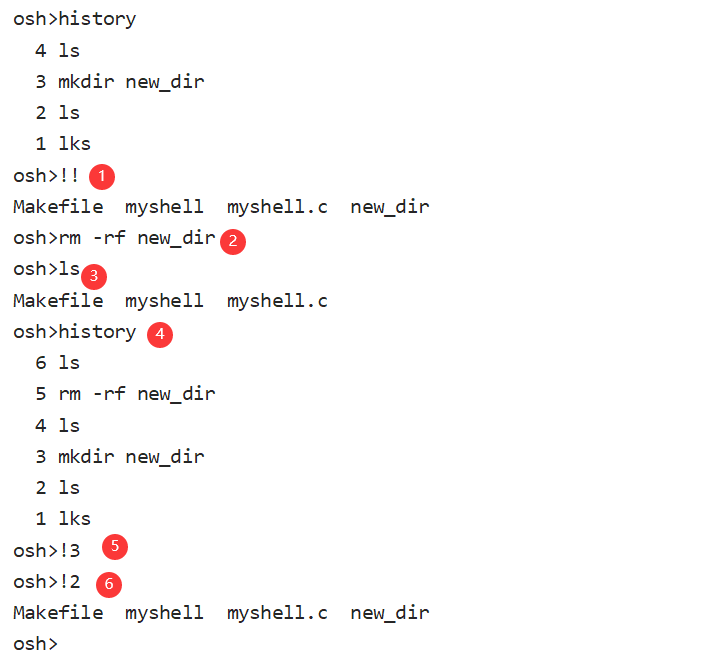
在上图 7 处后继续输入命令，如下图。

1 处为执行上一条命令，下一行即显示目录信息；

2 处删除刚才新建的文件夹，3 处可以看到文件夹被删除；

5 处为执行指定编号的命令，再次新建文件夹；

6 处为执行指定编号的命令，下一行可以看到新建的文件夹。



另外，程序也能对输入的正确性进行判断，如下图 1 和 2 处所示。

即历史命令中没有相关命令时会显示错误信息。

