

父 shell 与子 shell 与 shell 的子程序

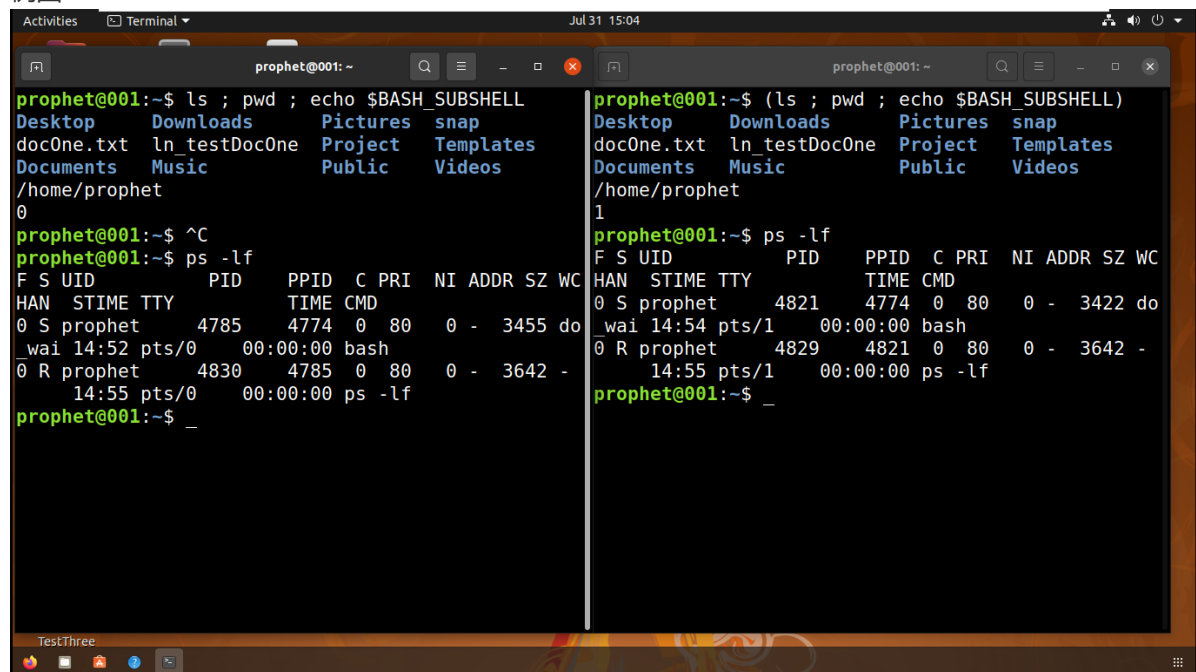
虽然子 Shell 的确是当前 Shell 的子进程，但当前 Shell 的子进程不一定是子 Shell（可能已经替换成了其他程序）。在 Bash 里面，只有特定的语法才会让代码进入子 Shell，比如管道两边的命令，比如用小括号括起来等等。真正的子 Shell 是不需要重新执行硬盘上的外部命令的，全部是内存中的操作。

演示

PID与UID与 & 与 ()

- PPID父进程id
- PID进程ID
- UID用户id
- &：将一个进程放置到后台运行的时候，Bash 会提示这个进程的进程 ID。
- () 包裹的命令：是当前 shell 中创建子 shell 并运行括号中的命令

例图：



```
prophet@001: ~  
prophet@001:~$ ls ; pwd ; echo $BASH_SUBSHELL  
Desktop Downloads Pictures snap  
docOne.txt ln_testDocOne Project Templates  
Documents Music Public Videos  
/home/prophet  
0  
prophet@001:~$ ^C  
prophet@001:~$ ps -lf  
F S UID PID PPID C PRI NI ADDR SZ WC  
HAN STIME TTY TIME CMD  
0 S prophet 4785 4774 0 80 0 - 3455 do  
_wai 14:52 pts/0 00:00:00 bash  
0 R prophet 4830 4785 0 80 0 - 3642 -  
14:55 pts/0 00:00:00 ps -lf  
prophet@001:~$ _  
  
prophet@001:~$ (ls ; pwd ; echo $BASH_SUBSHELL)  
Desktop Downloads Pictures snap  
docOne.txt ln_testDocOne Project Templates  
Documents Music Public Videos  
/home/prophet  
1  
prophet@001:~$ ps -lf  
F S UID PID PPID C PRI NI ADDR SZ WC  
HAN STIME TTY TIME CMD  
0 S prophet 4821 4774 0 80 0 - 3422 do  
_wai 14:54 pts/1 00:00:00 bash  
0 R prophet 4829 4821 0 80 0 - 3642 -  
14:55 pts/1 00:00:00 ps -lf  
prophet@001:~$ _
```

父与子之间的关系：

子Shell从父Shell继承得来的属性：

- 当前工作目录
- 部分环境变量
- 标准输入、标准输出和标准错误输出
- 所有已打开的文件标识符
- 忽略的信号

子Shell不能从父Shell继承的属性：

- 除环境变量和 .bashrc 文件中定义变量之外的Shell变量
- 未被忽略的信号处理

命令中; 的作用:

```
ls ; pwd ; cd /
```

利用分号隔开,可以在一条命令中,安排列顺序,从左至右依次执行命令

```
(ls ; pwd ; cd /)
```

创建子shell,并按上述方式执行命令

相关命令

echo输出指定的字符串或者变量

echo 多用于 shell 编程中,这里先不过多阐述.

选项

- e: 启用转义字符。
- E: 不启用转义字符（默认）
- n: 结尾不换行

目前常用命令

```
echo '输出的内容' #输出
```

```
echo $BASH_SUBSHELL #查看是否有子进程,并输出。
```

\$BASH_SUBSHELL 是 bash shell 变量:当前子进程数

sleep暂停指定的时间

时间长度, 后面可接 s、m、h 或 d, 其中 s 为秒, m 为分钟, h 为小时, d 为日数。

目前常用命令

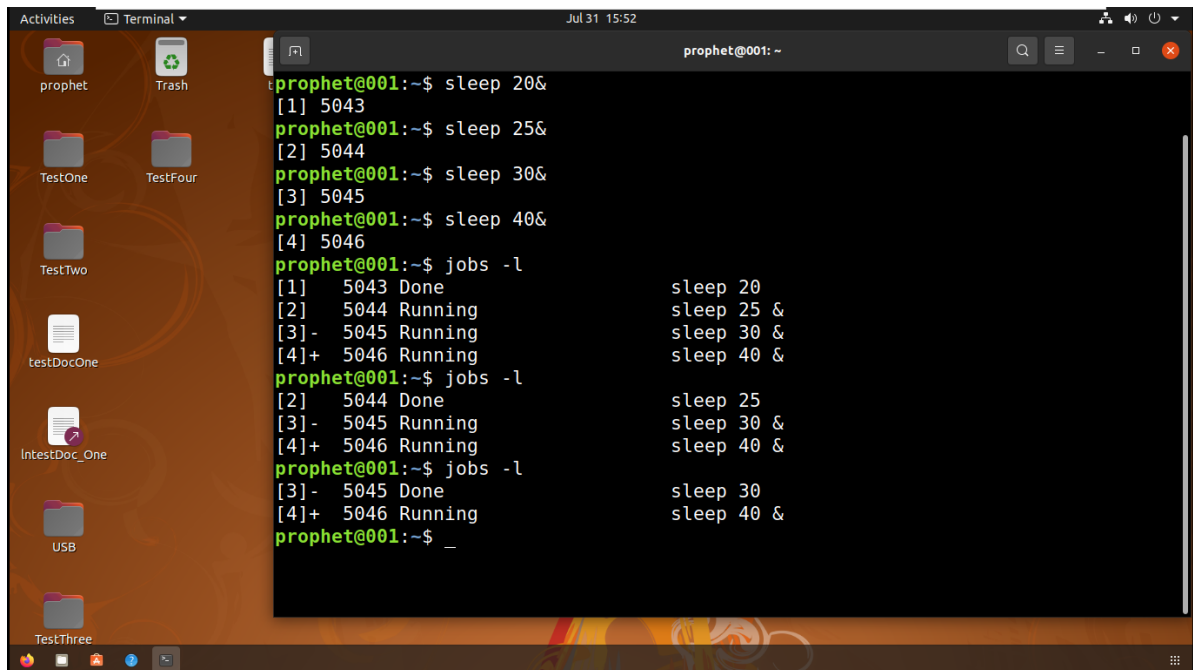
```
sleep 10      #睡眠10秒
sleep 300&    #放于后台执行300秒
```

jobs 可以用来查看当前终端放入后台的工作状态

选项

- l #在作业信息中额外的列出PID。
- n #只列出最近一次通知以来状态变更的作业。
- p #只列出PID。
- r #只输出处于运行状态的作业。
- s #只输出处于停止状态的作业。

例:



```
prophet@001:~$ sleep 20&
[1] 5043
prophet@001:~$ sleep 25&
[2] 5044
prophet@001:~$ sleep 30&
[3] 5045
prophet@001:~$ sleep 40&
[4] 5046
prophet@001:~$ jobs -l
[1] 5043 Done           sleep 20
[2] 5044 Running        sleep 25 &
[3]- 5045 Running        sleep 30 &
[4]+ 5046 Running        sleep 40 &
prophet@001:~$ jobs -l
[2] 5044 Done           sleep 25
[3]- 5045 Running        sleep 30 &
[4]+ 5046 Running        sleep 40 &
prophet@001:~$ jobs -l
[3]- 5045 Done           sleep 30
[4]+ 5046 Running        sleep 40 &
prophet@001:~$ _
```

+ 号代表最近一个放入后台的工作，也是工作恢复时默认恢复的工作。- 号代表倒数第二个放入后台的工作，而第三个以后的工作就没有 + 或 - 标志了。

coproc 协程处理命令

轻量级的线程，一个线程拥有多个协程

协程可以同时做两件事。在后台生成一个子shell，并在子shell中执行命令。

```
coproc sleep 10
coproc frank_av{ sleep 10; sleep 300;}
#每条命令一定要分号结尾，大括号内空格
```

alias 设置指令的别名

主要用途

- 简化较长的命令。
- 定义一个或多个别名。
- 修改一个或多个已定义别名的值。
- 显示一个或多个已定义别名。
- 显示全部已定义的别名。

选项

```
-p: 显示全部已定义的别名。
name (可选): 指定要 (定义、修改、显示) 的别名。
value (可选): 别名的值。
```

扩展

直接在shell里设定的命令别名，在终端关闭或者系统重新启动后都会失效，如何才能永久有效呢？

使用编辑器打开 `~/.bashrc`，在文件中加入别名设置，如：`alias rm='rm -i'` (别忘用单引号把要缩写的命令引起)，保存后执行 `source ~/.bashrc`，这样就可以永久保存命令的别名了。

因为修改的是当前用户目录下的 `~/.bashrc` 文件，所以这样的方式只对当前用户有用。如果要对所有用户都有效，修改 `/etc/bashrc` 文件就可以了。

type 显示指定命令的类型

该命令是bash内建命令。

外部命令 (built-in) 和内建命令 (external)

外部命令：

- 构建在 `shell` 外部,在系统加载时并不随系统一起被加载到内存中,而是在需要时才将其调用内存。
- 外部命令的执行 `shell` 进程会 `forking` (衍生)一个子进程,父进程随后挂起,然后在子进程中 `exec` (执行)加载外部文件,子进程返回后,父进程才继续执行,例如: `ps` `tar` 等
- 外部命令是在bash之外额外安装的,通常放在/bin, `/usr/bin`, `/sbin`, `/usr/sbin`...等等。

内建命令：

- 构建在`shell`内部,这些命令由 `shell` 程序识别并在shell程序内部完成运行,通常在 `Linux` 系统加载运行时 `shell` 就被加载并驻留在系统内存中。
- 写在 `bash` 源码里面,是一个独立的文件 (可以是二进制文件,也可以是一个脚本) 内建命令由当前 `shell` 本身来执行,例如 `echo`, `cd` 等。

选项

- a #在环境变量PATH中查找并显示所有包含name的可执行文件路径；当'-p'选项没有同时给出时，如果在别名、关键字，函数，内建的信息中存在name，则一并显示。
- f #排除对shell函数的查找。
- p #如果name在执行'type -t name'返回的不是'file'，那么什么也不返回；否则会在环境变量PATH中查找并返回可执行文件路径。
- P #即使要查找的name是别名、内建、函数中的一个，仍然会在环境变量PATH中查找并返回可执行文件路径。
- t #根据name的类型返回一个单词（别名，关键字，函数，内建，文件），否则返回空值。