

实验二 进程控制

1. 实验目的:

- 加深对进程概念的理解，明确进程和程序的区别。
- 掌握 Linux 系统中的进程创建，管理和删除等操作。
- 熟悉使用 Linux 下的命令和工具，如 man, find, grep, whereis, ps, pgrep, kill, ptree, top, vim, gcc, gdb, 管道|等。

2. 基础知识:

● 进程的创建

Linux 中，载入内存并执行程序映像的操作与创建一个新进程的操作是分离的。将程序映像载入内存，并开始运行它，这个过程称为运行一个新的程序，相应的系统调用称为 exec 系统调用。而创建一个新的进程的系统调用是 fork 系统调用。

● exec 系统调用

```
#include <unistd.h>

int execl (const char *path, const char *arg,...);
```

execl()将 path 所指路径的映像载入内存，arg 是它的第一个参数。参数可变长。参数列表必须以 NULL 结尾。

通常 execl()不会返回。成功的调用会以跳到新的程序入口点作为结束。发生错误时，execl()返回-1，并设置 errno 值。

例 编辑/home/kidd/hooks.txt:

```
int ret;

ret = execl ("/bin/vi", "vi","/home/kidd/hooks.txt", NULL);

if (ret == -1)

    perror ("execl");
```

● fork 系统调用

```
#include <sys/types.h>

#include <unistd.h>

pid_t fork (void);
```

成功调用 fork()会创建一个新的进程，它与调用 fork()的进程大致相同。发生错误时，fork()

返回-1，并设置 `errno` 值。

例：

```
pid_t pid;
pid = fork ();
if (pid > 0)
    printf ("I am the parent of pid=%d!\n", pid);
else if (!pid)
    printf ("I am the baby!\n");
else if (pid == -1)
    perror ("fork");
```

- 终止进程

`exit()`系统调用：

```
#include <stdlib.h>

void exit (int status);
```

- 进程挂起

`pause()` 系统调用：

```
int pause( void );
```

函数 `pause` 会把进程挂起，直到接收到信号。在信号接收后，进程会从 `pause` 函数中退出，继续运行。

- `wait()`(等待子进程中断或结束)

```
#include<sys/types.h>
#include<sys/wait.h>
pid_t wait (int * status);
```

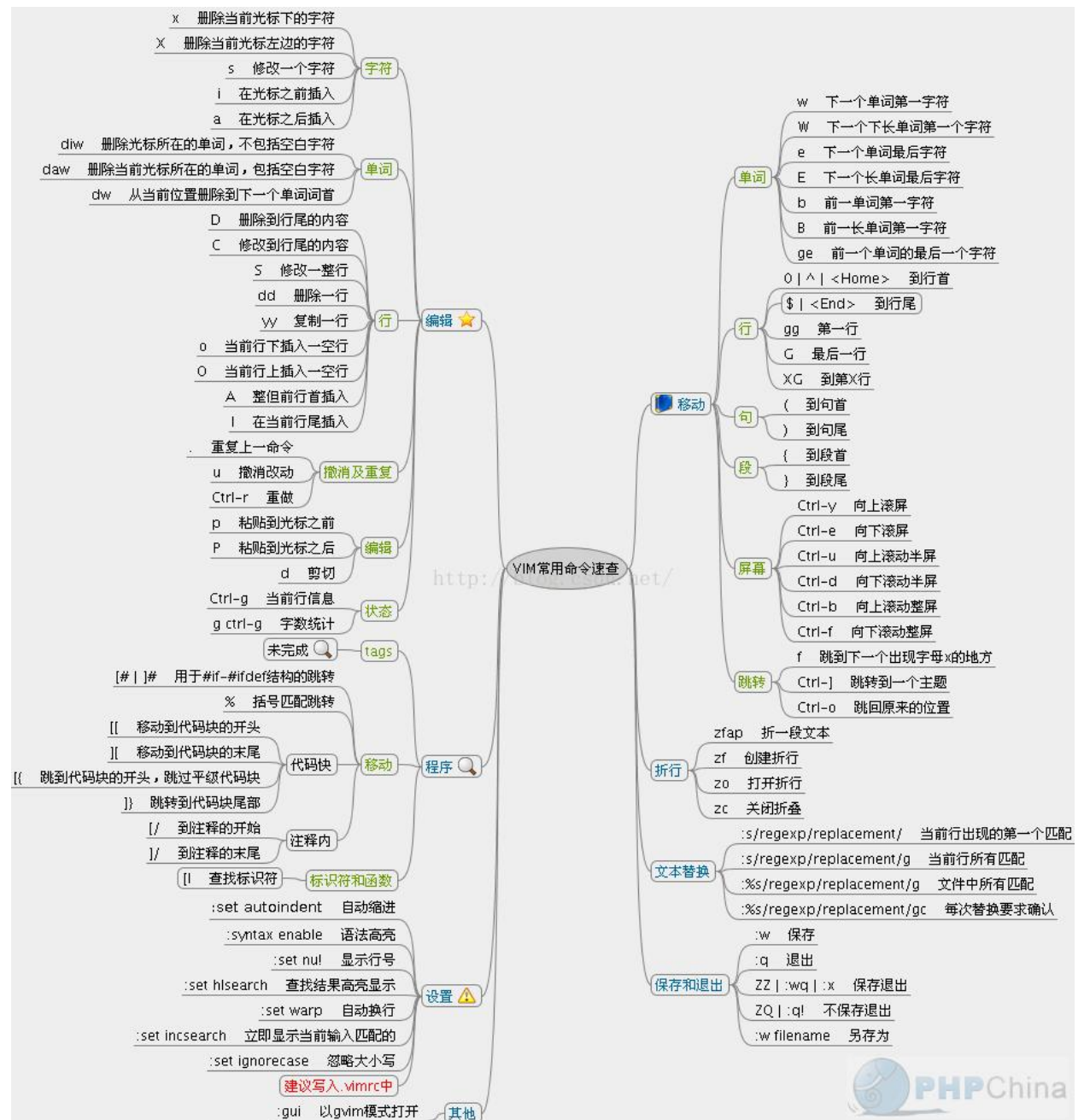
`wait()`会暂时停止目前进程的执行,直到有信号来到或子进程结束。

如果在调用 `wait()`时子进程已经结束,则 `wait()`会立即返回子进程结束状态值。

子进程的结束状态值会由参数 `status` 返回,而子进程的进程识别码也会一起返回。

如果不在意结束状态值,则参数 `status` 可以设成 `NULL`。

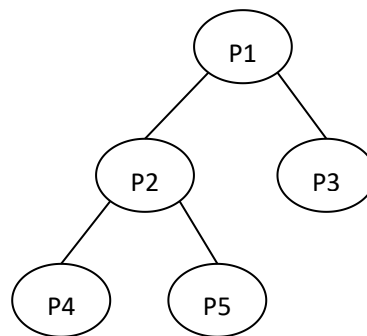
VIM 常用命令速查



3. 实验题目：

根据课堂所学内容和基础知识介绍，完成实验题目。

- 1、打开一个 `vi` 进程。通过 `ps` 命令以及选择合适的参数，只显示名字为 `vi` 的进程。寻找 `vi` 进程的父进程，直到 `init` 进程为止。记录过程中所有进程的 ID 和父进程 ID。将得到的进程树和由 `pstree` 命令的得到的进程树进行比较。
- 2、编写程序，首先使用 `fork` 系统调用，创建子进程。在父进程中继续执行空循环操作；在子进程中调用 `exec` 打开 `vi` 编辑器。然后在另外一个终端中，通过 `ps -Al` 命令、`ps aux` 或者 `top` 等命令，查看 `vi` 进程及其父进程的运行状态，理解每个参数所表达的意义。选择合适的命令参数，对所有进程按照 `cpu` 占用率排序。
- 3、使用 `fork` 系统调用，创建如下进程树，并使每个进程输出自己的 ID 和父进程的 ID。观察进程的执行顺序和运行状态的变化。



- 4、修改上述进程树中的进程，使得所有进程都循环输出自己的 ID 和父进程的 ID。然后终止 `p2` 进程(分别采用 `kill -9`、自己正常退出 `exit()`、段错误退出)，观察 `p1`、`p3`、`p4`、`p5` 进程的运行状态和其他相关参数有何改变。

4. 实验结果及分析

1、

打开一个 `vi` 进程。通过 `ps` 命令以及选择合适的参数，只显示名字为 `vi` 的进程。

```
smudge@smudge-ThinkPad-S2-2nd-Gen: ~
3407 ?      00:00:00 gvfsd-metadata
3412 ?      00:00:00 store
3415 ?      00:00:00 oosplash
3433 ?      00:00:36 soffice.bin
3529 ?      00:00:00 gvfsd-network
3557 ?      00:00:00 gvfsd-dnssd
3632 ?      00:00:00 kworker/u8:0
3634 ?      00:00:00 kworker/2:1
3637 ?      00:00:00 kworker/3:0
3641 ?      00:00:00 kworker/1:1
3678 ?      00:00:00 kworker/0:0
3723 ?      00:00:00 kworker/3:2
3724 ?      00:00:00 kworker/2:2
3733 ?      00:00:00 gnome-terminal-
3738 pts/1    00:00:00 bash
3749 pts/1    00:00:00 vi
3767 ?      00:00:00 kworker/0:1
3771 ?      00:00:00 kworker/1:0
3784 pts/2    00:00:00 bash
3796 pts/2    00:00:00 ps
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ ps -p3749
  PID TTY          TIME CMD
 3749 pts/1    00:00:00 vi
smudge@smudge-ThinkPad-S2-2nd-Gen:~$
```

如图所示，打开 vi 进程的进程号是 3749。

输入 `ps -ef|grep vi` 第二列为进程的 ID，第三列为父进程的 PID：

寻找 vi 进程的父进程，直到 init 进程为止。

```
smudge@smudge-ThinkPad-S2-2nd-Gen: ~
4    7    3262  3258  20    0    81244  5640  -    S    ?    0:00 /usr/lib/cu
1    0    3343    2    20    0    0      0      -    S    ?    0:00 [kworker/2:
0 1000  3407  1360  20    0 187716  5572  poll_s Sl  ?    0:00 /usr/lib/gv
4   105  3412  2212  25    5  52920 10908  -    SN  ?    0:00 /usr/lib/ap
0 1000  3415  1360  20    0 135868  5760  pipe_w Sl  ?    0:00 /usr/lib/li
0 1000  3433  3415  20    0 1534400 208868 poll_s Sl  ?    0:38 /usr/lib/li
0 1000  3529  1360  20    0 354672  7056  poll_s Sl  ?    0:00 /usr/lib/gv
0 1000  3557  1360  20    0 363456  6948  poll_s Sl  ?    0:00 /usr/lib/gv
1    0    3632    2    20    0    0      0      -    S    ?    0:00 [kworker/u8
1    0    3641    2    20    0    0      0      -    S    ?    0:00 [kworker/1:
1    0    3678    2    20    0    0      0      -    S    ?    0:00 [kworker/0:
1    0    3723    2    20    0    0      0      -    S    ?    0:00 [kworker/3:
1    0    3724    2    20    0    0      0      -    S    ?    0:00 [kworker/2:
0 1000  3733  1360  20    0 591396  36552 poll_s Rl  ?    0:01 /usr/lib/gn
0 1000  3738  3733  20    0  24168  5200  wait  Ss  pts/1 0:00 bash
0 1000  3749  3738  20    0  55368  8516  poll_s S+  pts/1 0:00 vi
1    0    3767    2    20    0    0      0      -    S    ?    0:00 [kworker/0:
1    0    3771    2    20    0    0      0      -    S    ?    0:00 [kworker/1:
0 1000  3784  3733  20    0  24168  5252  wait  Ss  pts/2 0:00 bash
1    0    3835    2    20    0    0      0      -    S    ?    0:00 [kworker/2:
1    0    3836    2    20    0    0      0      -    S    ?    0:00 [kworker/3:
4    0    3854    1    20    0 15676  1088  -    Ss  ?    0:00 /lib/system
4 1000  3857  3784  20    0  30656  1508  -    R+  pts/2 0:00 ps -lax
smudge@smudge-ThinkPad-S2-2nd-Gen:~$
```



```
smudge@smudge-ThinkPad-S2-2nd-Gen: ~  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ ps -ef|grep 3738  
smudge 3738 3733 0 08:22 pts/1 00:00:00 bash  
smudge 3749 3738 0 08:22 pts/1 00:00:00 vi  
smudge 4315 3784 0 08:54 pts/2 00:00:00 grep --color=auto 3738  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ ps -ef|grep 3733  
smudge 3733 1360 0 08:22 ? 00:00:07 /usr/lib/gnome-terminal/gnome-terminal-server  
smudge 3738 3733 0 08:22 pts/1 00:00:00 bash  
smudge 3784 3733 0 08:24 pts/2 00:00:00 bash  
smudge 4317 3784 0 08:55 pts/2 00:00:00 grep --color=auto 3733  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ ps -ef|grep 1360  
smudge 1360 1094 0 08:00 ? 00:00:00 /sbin/upstart --user  
smudge 1439 1360 0 08:00 ? 00:00:00 upstart-udev-bridge --daemon --user  
smudge 1450 1360 0 08:00 ? 00:00:02 dbus-daemon --fork --session --address=unix:abstract=/tmp/dbus-omz4JVXAWp  
smudge 1462 1360 0 08:00 ? 00:00:00 /usr/lib/x86_64-linux-gnu/hud/window-stack-bridge  
smudge 1487 1360 0 08:00 ? 00:00:02 /usr/bin/fcitx  
smudge 1503 1360 0 08:00 ? 00:00:01 /usr/bin/dbus-daemon --fork --print-pid 5 --print-address 7 --config-file /usr/share/fcitx/dbus/daemon.conf  
smudge 1511 1360 0 08:00 ? 00:00:00 /usr/bin/fcitx-dbus-watcher unix:abstract=/tmp/dbus-uKjsxNJgut,guid=aa70990b4e8712fadd3335335c8a0a16 1503  
smudge 1518 1360 0 08:00 ? 00:00:02 /usr/lib/x86_64-linux-gnu/bamf/b  
smudge@smudge-ThinkPad-S2-2nd-Gen: ~  
smudge 3407 1360 0 08:07 ? 00:00:00 /usr/lib/gvfs/gvfsd-metadata  
smudge 3415 1360 0 08:07 ? 00:00:00 /usr/lib/libreoffice/program/oosplash --writer file:///home/smudge/Downloads/%C3%8A%C2%B5%C3%91%C3%A9%C2%B6%C3%BE%20%C2%BD%C3%B8%C2%B3%C3%8C%C2%BF%C3%98%C3%96%C3%86-2019.docx  
smudge 3529 1360 0 08:14 ? 00:00:00 /usr/lib/gvfs/gvfsd-network --spawner :1.10 /org/gtk/gvfs/exec_spaw/1  
smudge 3557 1360 0 08:14 ? 00:00:00 /usr/lib/gvfs/gvfsd-dnssd --spawner :1.10 /org/gtk/gvfs/exec_spaw/5  
smudge 3733 1360 0 08:22 ? 00:00:07 /usr/lib/gnome-terminal/gnome-terminal-server  
smudge 4331 3784 0 08:55 pts/2 00:00:00 grep --color=auto 1360  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ ps -ef|grep 1094  
root 1094 980 0 08:00 ? 00:00:00 lightdm --session-child 12 19  
smudge 1360 1094 0 08:00 ? 00:00:00 /sbin/upstart --user  
smudge 4374 3784 0 08:58 pts/2 00:00:00 grep --color=auto 1094  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ ps -ef|grep 980  
root 980 1 0 08:00 ? 00:00:00 /usr/sbin/lightdm  
root 998 980 3 08:00 tty7 00:02:16 /usr/lib/xorg/Xorg -core :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch  
root 1094 980 0 08:00 ? 00:00:00 lightdm --session-child 12 19  
smudge 1980 1360 0 08:00 ? 00:00:00 /usr/lib/gvfs/gvfs-udisks2-volume-monitor  
smudge 4376 3784 0 08:58 pts/2 00:00:00 grep --color=auto 980  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$
```

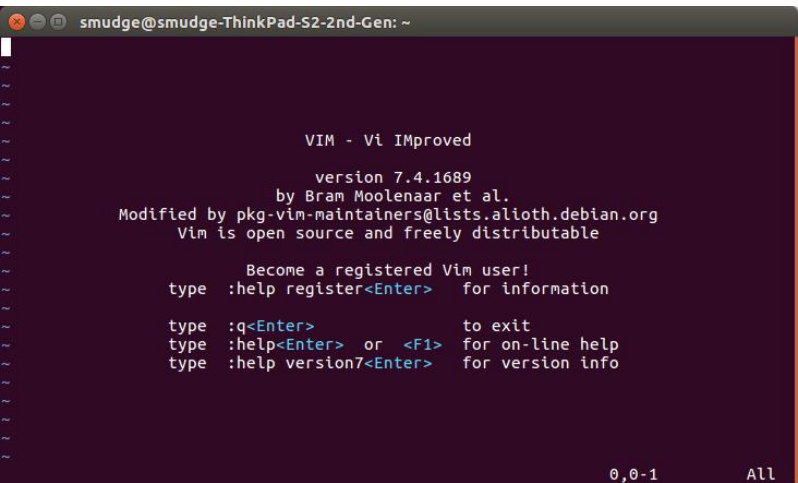
依次查询父进程，直到进程号为 1。可得父进程的顺序为：

3749->3738->3733->1360->1094->980->1

在 terminal 中输入 `ps tree -p` 查看进程树。

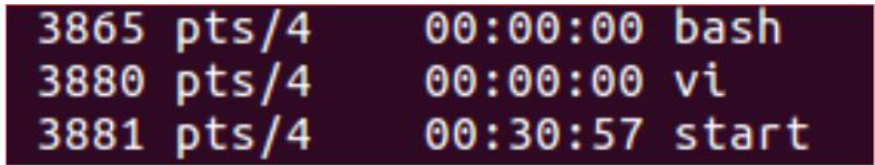
2、（代码见 [github](#)）

编写程序，首先使用 `fork` 系统调用，创建子进程。在父进程中继续执行空循环操作；在子进程中调用 `exec` 打开 `vi` 编辑器。

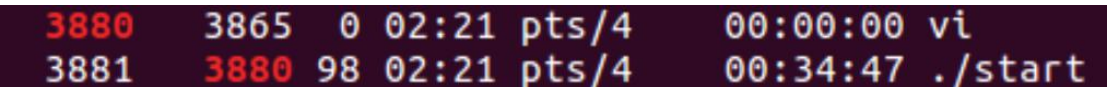


然后在另外一个终端中，通过 `ps -Al` 命令、`ps aux` 或者 `top` 等命令，查看 `vi` 进程及其父进程的运行状态，理解每个参数所表达的意义。选择合适的命令参数，对所有进程按照 `cpu` 占用率排序。

输入 `ps -A` 命令，得到 `vi` 的进程 ID 为 3880。



输入 `ps -ef|grep 3880` 查看父进程的进程 ID。



输入 `ps -lax` 查看进程参数：

F	UID	PID	PPID	PRI	NI	VSZ	RSS	WCHAN	STAT	TTY	TIME	COMMAND
0	1000	3880	3865	20	0	39108	3552	poll_s	S+	pts/4	0:00	vi
1	1000	3881	3880	20	0	4220	76	-	R+	pts/4	33:14	./start

可知 `vi` 进程状态在后台进程组中处于休眠状态，父进程为后台进程组中处于运行状态。

其中 `PID` 为 `vi` 进程的进程号。

`PPID` 为 `vi` 进程的父进程号。

`PRI` 是内核调度的优先级。

`NI` 是进程优先级。

`VSZ` 是总虚拟内存大小，以 `byte` 计。

RSS 是进程使用的总物理内存数，以 Kbytes 计。

STAT 为进程状态。

【D:不可终端；R: 正在运行或在队列中的进程；S:处于休眠状态；T: 停止或被追踪；

Z: 僵

尸进程；W: 进入内存交换；X: 死掉的进程；<:高优先级；N: 低优先级；L: 有些页被缩

进内存；s:包含子进程；+: 位于后台的进程组；|: 多线程】

TTY 为终端的次要装置码。

TIME 为使用 CPU 的时间。

输入 top，打开大写键盘，敲入 P。可得到按 CPU 占用率排序的所有进程：

```
top - 03:19:56 up 2:04, 1 user, load average: 1.42, 1.11, 1.03
Tasks: 224 total, 2 running, 222 sleeping, 0 stopped, 0 zombie
%Cpu(s): 99.3 us, 0.7 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 985856 total, 108040 free, 542464 used, 335352 buff/cache
KiB Swap: 1046524 total, 785916 free, 260608 used. 227508 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3881	li	20	0	4220	76	0	R	98.7	0.0	57:19.31	start
1027	root	20	0	433468	32636	15524	S	0.7	3.3	0:37.28	Xorg
2253	li	20	0	1223920	40188	24920	S	0.3	4.1	0:53.39	compiz
3860	li	20	0	596968	34012	25712	S	0.3	3.4	0:01.85	gnome-term+
1	root	20	0	119940	4108	2588	S	0.0	0.4	0:02.70	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd

3、（代码见 girhub——tree1.c）

运行结果如下所示：

```
smudge@smudge-ThinkPad-S2-2nd-Gen: ~
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ gcc -o tree1 tree1.c
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ ./tree1
p1 为根进程,pid = 4185
p2 是 p1 的子进程 pid = 4186, p2 的父进程为 ppid = 4185
p5 是 p2 的子进程 pid = 4187, p5 的父进程为 ppid = 4186
p4 是 p2 的子进程 pid = 4188, p4 的父进程为 ppid = 4186
p3 是 p1 的子进程 pid = 4189, p3 的父进程为 ppid = 4185
smudge@smudge-ThinkPad-S2-2nd-Gen:~$
```

可得 p1 进程 ID 为 4185。

p2 进程 ID 为 4186,父进程为 4185,即 p1.p3 进程 ID 为 4189,父进程为 4185,即 p1。

p4 进程 ID 为 4188,父进程为 4186,即 p2.p5 进程 ID 为 4187,父进程为 4186,即 p2。

首先创建的进程为 p1，然后是 p1 的子进程 p2，再然后是 p2 的子进程 p5，再是 p2 的子进程 p4，最后是 p1 的子进程 p3。

4、（代码见 [github——tree2.c/tree3.c/tree4.c](#)）

循环输出：

```
smudge@smudge-ThinkPad-S2-2nd-Gen: ~  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ ./tree2  
p5 是 p2 的子进程 pid = 4489, p5 的父进程为 ppid = 4488  
p5 是 p2 的子进程 pid = 4489, p5 的父进程为 ppid = 4488  
p3 是 p1 的子进程 pid = 4490, p3 的父进程为 ppid = 4487  
p4 是 p2 的子进程 pid = 4491, p4 的父进程为 ppid = 4488  
p5 是 p2 的子进程 pid = 4489, p5 的父进程为 ppid = 4488  
p3 是 p1 的子进程 pid = 4490, p3 的父进程为 ppid = 4487  
p1 为根进程, pid = 4487  
p2 是 p1 的子进程 pid = 4488, p2 的父进程为 ppid = 4487  
p4 是 p2 的子进程 pid = 4491, p4 的父进程为 ppid = 4488  
p1 为根进程, pid = 4487  
p2 是 p1 的子进程 pid = 4488, p2 的父进程为 ppid = 4487  
p3 是 p1 的子进程 pid = 4490, p3 的父进程为 ppid = 4487  
p5 是 p2 的子进程 pid = 4489, p5 的父进程为 ppid = 4488  
p4 是 p2 的子进程 pid = 4491, p4 的父进程为 ppid = 4488  
p5 是 p2 的子进程 pid = 4489, p5 的父进程为 ppid = 4488  
p4 是 p2 的子进程 pid = 4491, p4 的父进程为 ppid = 4488  
p1 为根进程, pid = 4487  
p3 是 p1 的子进程 pid = 4490, p3 的父进程为 ppid = 4487  
p2 是 p1 的子进程 pid = 4488, p2 的父进程为 ppid = 4487  
p5 是 p2 的子进程 pid = 4489, p5 的父进程为 ppid = 4488  
p4 是 p2 的子进程 pid = 4491, p4 的父进程为 ppid = 4488  
p1 为根进程, pid = 4487  
p3 是 p1 的子进程 pid = 4490, p3 的父进程为 ppid = 4487
```

（1）输入 `kill -9 4488`，关闭 ID 为 4488 的进程，即 p2。可看到 p4 和 p5 的父进程成为 p1 的父进程。

```
smudge@smudge-ThinkPad-S2-2nd-Gen: ~  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ kill -9 4488  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$
```

```
smudge@smudge-ThinkPad-S2-2nd-Gen: ~  
p1 为根进程, pid = 4487  
p4 是 p2 的子进程 pid = 4491, p4 的父进程为 ppid = 1332  
p3 是 p1 的子进程 pid = 4490, p3 的父进程为 ppid = 4487  
p1 为根进程, pid = 4487  
p3 是 p1 的子进程 pid = 4490, p3 的父进程为 ppid = 4487  
p5 是 p2 的子进程 pid = 4489, p5 的父进程为 ppid = 1332  
p4 是 p2 的子进程 pid = 4491, p4 的父进程为 ppid = 1332  
p5 是 p2 的子进程 pid = 4489, p5 的父进程为 ppid = 1332  
p1 为根进程, pid = 4487  
p4 是 p2 的子进程 pid = 4491, p4 的父进程为 ppid = 1332  
p3 是 p1 的子进程 pid = 4490, p3 的父进程为 ppid = 4487  
p5 是 p2 的子进程 pid = 4489, p5 的父进程为 ppid = 1332  
p1 为根进程, pid = 4487  
p4 是 p2 的子进程 pid = 4491, p4 的父进程为 ppid = 1332  
p3 是 p1 的子进程 pid = 4490, p3 的父进程为 ppid = 4487  
p5 是 p2 的子进程 pid = 4489, p5 的父进程为 ppid = 1332  
p1 为根进程, pid = 4487  
p4 是 p2 的子进程 pid = 4491, p4 的父进程为 ppid = 1332  
p3 是 p1 的子进程 pid = 4490, p3 的父进程为 ppid = 4487  
p5 是 p2 的子进程 pid = 4489, p5 的父进程为 ppid = 1332
```


输入 `ps -lax` 查看各进程的运行状态。

```
smudge@smudge-ThinkPad-S2-2nd-Gen: ~  
0 1000 3933 1332 20 0 616404 48180 poll_s SL ? 0:08 /usr/lib/gn  
0 1000 3938 3933 20 0 24068 5144 wait_w Ss+ pts/2 0:00 bash  
1 0 3989 2 20 0 0 0 - S ? 0:00 [kworker/2:  
1 0 4004 2 20 0 0 0 - S ? 0:00 [kworker/3:  
0 1000 4026 3933 20 0 24068 5144 wait_w Ss+ pts/6 0:00 bash  
1 0 4070 2 20 0 0 0 - S ? 0:00 [kworker/2:  
1 0 4258 2 20 0 0 0 - S ? 0:00 [kworker/0:  
1 0 4297 2 20 0 0 0 - S ? 0:00 [kworker/3:  
1 0 4300 2 20 0 0 0 - S ? 0:00 [kworker/1:  
1 0 4301 2 20 0 0 0 - S ? 0:00 [kworker/u8  
1 0 4510 2 20 0 0 0 - S ? 0:00 [kworker/2:  
0 1000 4570 3933 20 0 24068 4824 wait Ss pts/9 0:00 bash  
0 1000 4582 4570 20 0 4352 620 hrttime S+ pts/9 0:00 ./tree2  
1 1000 4583 4582 20 0 4352 72 hrttime S+ pts/9 0:00 ./tree2  
1 1000 4584 4583 20 0 4352 76 hrttime S+ pts/9 0:00 ./tree2  
1 1000 4585 4582 20 0 4352 72 hrttime S+ pts/9 0:00 ./tree2  
1 1000 4586 4583 20 0 0 0 - Z+ pts/9 0:00 [tree2] <de  
1 0 4604 2 20 0 0 0 - S ? 0:00 [kworker/0:  
1 0 4623 2 20 0 0 0 - S ? 0:00 [kworker/3:  
0 1000 4638 3933 20 0 24068 4824 wait Ss pts/20 0:00 bash  
1 0 4648 2 20 0 0 0 - S ? 0:00 [kworker/1:  
4 0 4662 1 20 0 15676 1128 - Ss ? 0:00 /lib/system  
4 1000 4680 4638 20 0 30656 1624 - R+ pts/20 0:00 ps -lax  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$
```

VSZ 为总虚拟内存大小, p2 进程虚拟内存变为 0。P2 状态为退出状态, 进程称为僵尸进程。
p1, p3, p4, p5 进程为可中断的睡眠状态。

(2) exit(0)退出

在 p2 进程中插入 `exit(0)`语句, 关闭进程。可看到 p4 和 p5 的父进程为 p1 的父进程。

```
smudge@smudge-ThinkPad-S2-2nd-Gen: ~  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ gcc -o tree3 tree3.c  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ ./tree3  
p5 是 p2 的子进程 pid = 4837, p5 的父进程为 ppid = 4836  
p5 是 p2 的子进程 pid = 4837, p5 的父进程为 ppid = 4836  
p4 是 p2 的子进程 pid = 4839, p4 的父进程为 ppid = 4836  
p3 是 p1 的子进程 pid = 4840, p3 的父进程为 ppid = 4835  
p5 是 p2 的子进程 pid = 4837, p5 的父进程为 ppid = 4836  
p1 为根进程, pid = 4835  
p4 是 p2 的子进程 pid = 4839, p4 的父进程为 ppid = 1332  
p3 是 p1 的子进程 pid = 4840, p3 的父进程为 ppid = 4835  
p5 是 p2 的子进程 pid = 4837, p5 的父进程为 ppid = 1332  
p1 为根进程, pid = 4835  
p3 是 p1 的子进程 pid = 4840, p3 的父进程为 ppid = 4835  
p4 是 p2 的子进程 pid = 4839, p4 的父进程为 ppid = 1332  
p5 是 p2 的子进程 pid = 4837, p5 的父进程为 ppid = 1332  
p1 为根进程, pid = 4835  
p3 是 p1 的子进程 pid = 4840, p3 的父进程为 ppid = 4835  
p4 是 p2 的子进程 pid = 4839, p4 的父进程为 ppid = 1332  
p5 是 p2 的子进程 pid = 4837, p5 的父进程为 ppid = 1332  
p1 为根进程, pid = 4835  
p3 是 p1 的子进程 pid = 4840, p3 的父进程为 ppid = 4835  
p4 是 p2 的子进程 pid = 4839, p4 的父进程为 ppid = 1332  
p5 是 p2 的子进程 pid = 4837, p5 的父进程为 ppid = 1332  
p1 为根进程, pid = 4835
```

输入 `ps -lax` 查看各进程的运行状态:

```
smudge@smudge-ThinkPad-S2-2nd-Gen: ~  
0 1000 3938 3933 20 0 24068 5144 wait_w Ss+ pts/2 0:00 bash  
1 0 3989 2 20 0 0 0 - S ? 0:00 [kworker/2:  
0 1000 4026 3933 20 0 24068 5144 wait Ss pts/6 0:00 bash  
1 0 4070 2 20 0 0 0 - S ? 0:00 [kworker/2:  
1 0 4297 2 20 0 0 0 - S ? 0:00 [kworker/3:  
1 0 4510 2 20 0 0 0 - S ? 0:00 [kworker/2:  
0 1000 4722 1332 20 0 681912 47180 poll_s Sl ? 0:09 gedit /home  
1 0 4738 2 20 0 0 0 - S ? 0:00 [kworker/0:  
1 0 4739 2 20 0 0 0 - S ? 0:00 [kworker/3:  
1 0 4742 2 20 0 0 0 - S ? 0:00 [kworker/1:  
1 0 4749 2 20 0 0 0 - S ? 0:00 [kworker/u8  
0 1000 4764 1332 20 0 750584 45572 poll_s Sl ? 0:01 file-roller  
1 0 4814 2 20 0 0 0 - S ? 0:00 [kworker/0:  
1 0 4821 2 20 0 0 0 - S ? 0:00 [kworker/3:  
0 1000 4835 4026 20 0 4352 620 hrttime S+ pts/6 0:00 ./tree3  
1 1000 4836 4835 20 0 0 0 - Z+ pts/6 0:00 [tree3] <de  
1 1000 4837 1332 20 0 4352 76 hrttime S+ pts/6 0:00 ./tree3  
1 0 4838 2 20 0 0 0 - S ? 0:00 [kworker/1:  
1 1000 4839 1332 20 0 4352 76 hrttime S+ pts/6 0:00 ./tree3  
1 1000 4840 4835 20 0 4352 72 hrttime S+ pts/6 0:00 ./tree3  
4 0 4853 1 20 0 15676 1128 - Ss ? 0:00 /lib/system  
0 1000 4867 3933 20 0 24068 4824 wait Ss pts/9 0:00 bash  
4 1000 4877 4867 20 0 30656 1624 - R+ pts/9 0:00 ps -lax  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$
```

结果与 kill-9 得到的结果相同。

(3) 段错误退出

在 p2 进程中插入语句: int *ptr = NULL;*ptr = 0;关闭进程。可看到 p4 和 p5 的父进程为 p1 的父进程。

```
smudge@smudge-ThinkPad-S2-2nd-Gen: ~  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ gcc -o tree4 tree4.c  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$ ./tree4  
p5 是 p2 的子进程 pid = 4985, p5 的父进程为 ppid = 4984  
p5 是 p2 的子进程 pid = 4985, p5 的父进程为 ppid = 4984  
p4 是 p2 的子进程 pid = 4986, p4 的父进程为 ppid = 4984  
p3 是 p1 的子进程 pid = 4987, p3 的父进程为 ppid = 4983  
p5 是 p2 的子进程 pid = 4985, p5 的父进程为 ppid = 4984  
p1 为根进程, pid = 4983  
p4 是 p2 的子进程 pid = 4986, p4 的父进程为 ppid = 4984  
p3 是 p1 的子进程 pid = 4987, p3 的父进程为 ppid = 4983  
p5 是 p2 的子进程 pid = 4985, p5 的父进程为 ppid = 1332  
p1 为根进程, pid = 4983  
p4 是 p2 的子进程 pid = 4986, p4 的父进程为 ppid = 1332  
p3 是 p1 的子进程 pid = 4987, p3 的父进程为 ppid = 4983  
p1 为根进程, pid = 4983  
p4 是 p2 的子进程 pid = 4986, p4 的父进程为 ppid = 1332  
p3 是 p1 的子进程 pid = 4987, p3 的父进程为 ppid = 4983  
p5 是 p2 的子进程 pid = 4985, p5 的父进程为 ppid = 1332  
p1 为根进程, pid = 4983  
p5 是 p2 的子进程 pid = 4985, p5 的父进程为 ppid = 1332  
p4 是 p2 的子进程 pid = 4986, p4 的父进程为 ppid = 1332  
p3 是 p1 的子进程 pid = 4987, p3 的父进程为 ppid = 4983  
p4 是 p2 的子进程 pid = 4986, p4 的父进程为 ppid = 1332  
p1 为根进程, pid = 4983
```


输入 `ps -lax` 查看各进程的运行状态，结果与（1）（2）相同。

```
smudge@smudge-ThinkPad-S2-2nd-Gen: ~  
0 1000 3865 1332 20 0 178496 4072 poll_s Sl ? 0:00 /usr/lib/ev  
1 0 3989 2 20 0 0 0 - S ? 0:00 [kworker/2:  
1 0 4070 2 20 0 0 0 - S ? 0:00 [kworker/2:  
1 0 4297 2 20 0 0 0 - S ? 0:00 [kworker/3:  
1 0 4510 2 20 0 0 0 - S ? 0:00 [kworker/2:  
1 0 4739 2 20 0 0 0 - S ? 0:00 [kworker/3:  
1 0 4742 2 20 0 0 0 - S ? 0:00 [kworker/1:  
1 0 4814 2 20 0 0 0 - S ? 0:00 [kworker/0:  
1 0 4821 2 20 0 0 0 - S ? 0:00 [kworker/3:  
1 0 4838 2 20 0 0 0 - S ? 0:00 [kworker/1:  
1 0 4900 2 20 0 0 0 - S ? 0:00 [kworker/u8  
1 0 4901 2 20 0 0 0 - S ? 0:00 [kworker/2:  
1 0 4940 2 20 0 0 0 - S ? 0:00 [kworker/u8  
0 1000 4963 1332 20 0 610644 42664 poll_s Sl ? 0:00 /usr/lib/gn  
0 1000 4968 4963 20 0 24068 5144 wait Ss pts/2 0:00 bash  
0 1000 4983 4968 20 0 4352 620 hrttime S+ pts/2 0:00 ./tree4  
1 1000 4984 4983 20 0 0 0 - Z+ pts/2 0:00 [tree4] <de  
1 1000 4985 1332 20 0 4352 76 hrttime S+ pts/2 0:00 ./tree4  
1 1000 4986 1332 20 0 4352 76 hrttime S+ pts/2 0:00 ./tree4  
1 1000 4987 4983 20 0 4352 72 hrttime S+ pts/2 0:00 ./tree4  
4 0 5001 1 20 0 15676 1128 - Ss ? 0:00 /lib/system  
0 1000 5023 4963 20 0 24068 4824 wait Ss pts/6 0:00 bash  
4 1000 5033 5023 20 0 30656 1624 - R+ pts/6 0:00 ps -lax  
smudge@smudge-ThinkPad-S2-2nd-Gen:~$
```