

实验二 进程控制

李许增 16281042

操作系统 arch Linux

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。

单步 ps 命令选择得到的进程树和用 pstree 命令得到的进程树相同。

ps -f -C name 命令为筛选指定名称进程的信息；

ps -p 进程号 -o comm= 为查找 pid 对应进程的名称；

ps -p -s 进程号 命令为查找指定进程号的进程树。

```
实验二: zsh - Konsole
文件(F) 编辑(E) 视图(V) 书签(B) 设置(S) 帮助(H)
liuxuzeng@xuzeng-Li ~/OS-2019/实验二 ps -f -C vi 408 13:38:11
UID PID PPID C STIME TTY TIME CMD
liuxuzeng 7401 844 0 13:34 pts/0 00:00:00 vi test
liuxuzeng@xuzeng-Li ~/OS-2019/实验二 ps -p 7401 -o comm= 409 13:38:13
vi
liuxuzeng@xuzeng-Li ~/OS-2019/实验二 ps -p 844 -o comm= 410 13:38:34
zsh
liuxuzeng@xuzeng-Li ~/OS-2019/实验二 ps -f -C zsh 411 13:38:44
UID PID PPID C STIME TTY TIME CMD
liuxuzeng 844 786 0 12:58 pts/0 00:00:00 /bin/zsh
liuxuzeng 4437 786 0 13:13 pts/2 00:00:00 /bin/zsh
liuxuzeng 7445 7441 0 13:35 pts/3 00:00:00 /bin/zsh
liuxuzeng@xuzeng-Li ~/OS-2019/实验二 ps -p 786 -o comm= 412 13:38:59
yakuake
liuxuzeng@xuzeng-Li ~/OS-2019/实验二 ps -f -C yakuake 413 13:39:20
UID PID PPID C STIME TTY TIME CMD
liuxuzeng 786 651 1 12:58 ? 00:00:30 /usr/bin/yakuake -session 1014fdca7b1000155252831700000006740033_15
liuxuzeng@xuzeng-Li ~/OS-2019/实验二 ps -p 651 -o comm= 414 13:39:30
kdeinit5
liuxuzeng@xuzeng-Li ~/OS-2019/实验二 ps -f -C kdeinit5 415 13:39:40
UID PID PPID C STIME TTY TIME CMD
liuxuzeng 651 1 0 12:58 ? 00:00:00 kdeinit5: Running...
liuxuzeng@xuzeng-Li ~/OS-2019/实验二 ps -p 1 -o comm= 416 13:39:56
systemd
liuxuzeng@xuzeng-Li ~/OS-2019/实验二 417 13:40:07
```

```
liuxuzeng@xuzeng-Li ~/OS-2019/实验二 pstree -p -s 7401 418 13:42:58
systemd(1)---kdeinit5(651)---yakuake(786)---zsh(844)---vi(7401)
liuxuzeng@xuzeng-Li ~/OS-2019/实验二 419 13:43:26
```

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。

程序代码：

```
#include<unistd.h>
#include<stdio.h>
int main()
{
    pid_t fork_pid;//fork_pid 为 fork 函数返回值
    fork_pid = fork();
    if(fork_pid < 0)//当返回值为负值时表示出现错误
    {
        printf("fork error");
    }
}
```

```

else if(fork_pid == 0)//当返回值为0时表示创建的子进程在运行
{
    printf("我是子进程, 我的 pid 为: %d\n", getpid());
    if((execlp("vi", "vi", "/home/os-2019/test", NULL))<0) //注意 execlp 参数使用
        printf("execlp error\n");
}
else//当返回值为其他正值时表示父进程在运行
{
    printf("我是父进程, 我的 pid 为: %d\n", getpid());
    while(1) {};
}
return 0;
}

```

首先利用 `ps tree` 命令查看 `vi` 进程以及父进程:

```

lixuzeng@xuzeng-Li ~/OS-2019/实验二/2$ ps -f -C vi
UID          PID    PPID  C STIME TTY          TIME CMD
lixuzeng 17611 17610  0 15:06 pts/4    00:00:00 vi /home/os-2019/test
lixuzeng@xuzeng-Li ~/OS-2019/实验二/2$ ps -p 17610 -o comm=
2_fork
lixuzeng@xuzeng-Li ~/OS-2019/实验二/2$ ps -f -C 2_fork
UID          PID    PPID  C STIME TTY          TIME CMD
lixuzeng 17610 15441 99 15:06 pts/4    00:01:56 ./2_fork
lixuzeng@xuzeng-Li ~/OS-2019/实验二/2$ ps -p 15441 -o comm=
zsh
lixuzeng@xuzeng-Li ~/OS-2019/实验二/2$ ps -f -C zsh
UID          PID    PPID  C STIME TTY          TIME CMD
lixuzeng 12576 12572  0 14:27 pts/2    00:00:00 /bin/zsh
lixuzeng 15441  786   0 14:48 pts/4    00:00:00 /bin/zsh
lixuzeng 16141 16137  0 14:53 pts/0    00:00:00 /bin/zsh
lixuzeng@xuzeng-Li ~/OS-2019/实验二/2$ ps -p 786 -o comm=
yakuake
lixuzeng@xuzeng-Li ~/OS-2019/实验二/2$ ps -f -C yakuake
UID          PID    PPID  C STIME TTY          TIME CMD
lixuzeng  786    651   0 12:58 ?        00:01:05 /usr/bin/yakuake -session 1014fdca7b100015525283170000006740033_15
lixuzeng@xuzeng-Li ~/OS-2019/实验二/2$ ps -p 651 -o comm=
kdeinit5
lixuzeng@xuzeng-Li ~/OS-2019/实验二/2$ ps -f -C kdeinit5
UID          PID    PPID  C STIME TTY          TIME CMD
lixuzeng  651      1   0 12:58 ?        00:00:00 kdeinit5: Running...
lixuzeng@xuzeng-Li ~/OS-2019/实验二/2$ pstree -p -s 17611
systemd(1)─kdeinit5(651)─yakuake(786)─zsh(15441)─2_fork(17610)─vi(17611)
lixuzeng@xuzeng-Li ~/OS-2019/实验二/2$

```

执行 `ps aux` 命令查看当前系统中运行程序情况:

制表项内容说明:

USER: 进程拥有者

PID: 进程 pid

%CPU: 占用的 CPU 使用率

%MEM: 占用的内存使用率

VSZ: 占用的虚拟内存大小

RSS: 占用的内存大小

TTY: 终端的次要装置号码 (minor device number of tty)

STAT: 该进程的状态, linux 的进程有以下状态:

D 不可中断 Uninterruptible (usually IO)

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 (从内核 2.6 开始无效)

X 死掉的进程

- < 高优先级
- n 低优先级
- s 包含子进程
- + 位于后台的进程组

START: 进程开始时间

TIME: 执行的时间

COMMAND: 对应执行的指令

```
2: zsh - Konsole
文件(F) 编辑(E) 视图(V) 书签(B) 设置(S) 帮助(H)
liuxuzeng@xuzeng-Li ~/OS-2019/实验二/2 ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 201468 9304 ?        Ss   12:58   0:00 /sbin/init \\boot\\vmlinuz-linux
root         2  0.0  0.0      0     0 ?        S    12:58   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I<   12:58   0:00 [rcu_gp]
root         4  0.0  0.0      0     0 ?        I<   12:58   0:00 [rcu_par_gp]
root         6  0.0  0.0      0     0 ?        I<   12:58   0:00 [kworker/0:0H-kblockd]
root         8  0.0  0.0      0     0 ?        I<   12:58   0:00 [mm_percpu_wq]
root         9  0.0  0.0      0     0 ?        S    12:58   0:00 [ksoftirqd/0]
root        10  0.0  0.0      0     0 ?        I    12:58   0:03 [rcu_preempt]
root        11  0.0  0.0      0     0 ?        I    12:58   0:00 [rcu_sched]
root        12  0.0  0.0      0     0 ?        I    12:58   0:00 [rcu_bh]
root        13  0.0  0.0      0     0 ?        S    12:58   0:00 [rcuc/0]
root        14  0.0  0.0      0     0 ?        S    12:58   0:00 [rcub/0]
root        15  0.0  0.0      0     0 ?        S    12:58   0:00 [migration/0]
root        16  0.0  0.0      0     0 ?        S    12:58   0:00 [idle_inject/0]
root        17  0.0  0.0      0     0 ?        I    12:58   0:01 [kworker/0:1-events]
root        18  0.0  0.0      0     0 ?        S    12:58   0:00 [cpuhp/0]
root        19  0.0  0.0      0     0 ?        S    12:58   0:00 [cpuhp/1]
root        20  0.0  0.0      0     0 ?        S    12:58   0:00 [idle_inject/1]
root        21  0.0  0.0      0     0 ?        S    12:58   0:00 [migration/1]
root        22  0.0  0.0      0     0 ?        S    12:58   0:00 [rcuc/1]
root        23  0.0  0.0      0     0 ?        S    12:58   0:00 [ksoftirqd/1]
root        25  0.0  0.0      0     0 ?        I<   12:58   0:00 [kworker/1:0H-kblockd]
root        26  0.0  0.0      0     0 ?        S    12:58   0:00 [cpuhp/2]
root        27  0.0  0.0      0     0 ?        S    12:58   0:00 [idle_inject/2]
```

对应和 vi 进程相关的进程信息:

```
liuxuzeng 651 0.0 0.2 121900 20012 ?        Ss   12:58   0:00 kdeinit5: Running...
liuxuzeng 786 0.8 1.0 480344 87320 ?        Sl   12:58   1:05 /usr/bin/yakuake -session 1014fdca7b10001552528317
liuxuzeng 15441 0.0 0.0 14352 7224 pts/4    Ss   14:48   0:00 /bin/zsh
liuxuzeng 17610 99.9 0.0 2296 748 pts/4    R+   15:06   5:44 ./2_fork
liuxuzeng 17611 0.0 0.0 9172 2616 pts/4    S+   15:06   0:00 vi /home/os-2019/test
```

ps -Al 命令查看进程信息:

```
liuxuzeng@xuzeng-Li ~/OS-2019/实验二/2 ps -Al
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY      TIME CMD
4 S  0 1 0 0 80 0 - 50367 - ? 00:00:00 systemd
1 S  0 2 0 0 80 0 - 0 - ? 00:00:00 kthreadd
1 I  0 3 2 0 60 -20 - 0 - ? 00:00:00 rcu_gp
1 I  0 4 2 0 60 -20 - 0 - ? 00:00:00 rcu_par_gp
```

制表项内容说明:

UID: 用户 ID (User ID)

PPID: 父进程 pid

S: 进程或内核线程的状态:

- O: 不存在
- A: 活动
- W: 已交换
- I: 空闲 (等待启动)
- Z: 已取消
- T: 已停止
- S: 正在休眠
- R: 正在运行

C: (-f、l 和 -l 标志) 每次系统时钟周期和发现线程或进程需要运行时增加进程或线程的 CPU 利用率。调度程序通过每秒将该值除以 2 一次来使其衰减。对于 sched_other 策略, CPU 利用率用于确定进程调度优先级。如果值较大, 那么表示一个将耗用大量 CPU 资源的进程, 该进程的优先级将更低; 如果值较小, 那么表示一个要执行大量 I/O 操作的进程, 其优先级将更高。

PRI: (-l 和 l 标志) 进程或内核线程的优先级; 数字越大优先级越低。

WCHAN: (-l 标志) 进程或内核线程为之等待或休眠的事件。对于内核线程, 如果内核线程正在运行, 该字段为空。对于进程, 如果只有一个内核线程正在休眠, 等待通道定义为该休眠内核线程的等待通道; 否则显示一个星号。

NI: (-l 和 l 标志) 为 sched other 策略计算优先级中使用的细调值。

ADDR: 通常情况下, (-l 和 l 标志) 包含 进程栈的段号; 如果为内核进程, 那么为预处理数据区的地址。

SZ: (-l 和 l 标志) 进程的核心映像大小 (以 1 KB 为单位)。

TIME: (所有标志) 进程的运行时间总和。如果运行时间达到 100 分钟, 以 mm:ss 或 mmmm:ss 格式显示时间, 这与使用 -o time 标志时的显示格式不同。

对应和 vi 进程相关的进程信息:

```
0 S 1000 786 651 0 80 0 - 120086 - ? 00:01:06 yakuake
0 S 1000 15441 786 0 80 0 - 3588 - pts/4 00:00:00 zsh
0 R 1000 17610 15441 99 80 0 - 574 - pts/4 00:15:35 2_fork
0 S 1000 17611 17610 0 80 0 - 2293 - pts/4 00:00:00 vi
```

top 命令实时显示系统中各个进程的资源占用状况:

```
2: top - Konsole
文件(F) 编辑(E) 视图(V) 书签(B) 设置(S) 帮助(H)
1 I 0 19634 2 0 80 0 - 0 - ? 00:00:00 kworker/1:3
1 I 0 19635 2 0 80 0 - 0 - ? 00:00:00 kworker/3:2
1 I 0 19649 2 0 80 0 - 0 - ? 00:00:00 kworker/6:0-events
1 I 0 19650 2 0 80 0 - 0 - ? 00:00:00 kworker/0:2
0 R 1000 19651 12576 0 80 0 - 4437 - pts/2 00:00:00 ps
lixuzeng@xuzeng-Li ~ /OS-2019/实验二/2 top 498 15:21:43
top - 15:27:11 up 2:29, 5 users, load average: 2.26, 2.27, 1.79
任务: 263 total, 2 running, 260 sleeping, 0 stopped, 1 zombie
%Cpu(s): 14.4 us, 1.9 sy, 0.0 ni, 83.5 id, 0.0 wa, 0.1 hi, 0.1 si, 0.0 st
MiB Mem : 7848.5 total, 3068.3 free, 2545.7 used, 2234.6 buff/cache
MiB Swap: 5120.0 total, 5120.0 free, 0.0 used, 4613.3 avail Mem

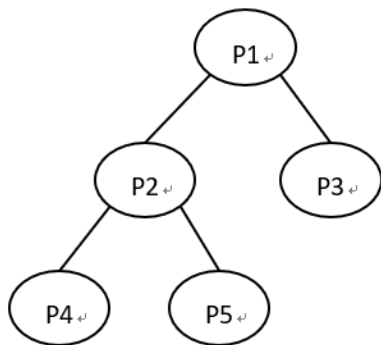
 进程 USER      PR    NI    VIRT    RES    SHR    %CPU    %MEM    TIME+  COMMAND
17610 lixuzeng 20     0    2296     748     684 R 100.0    0.0   21:02.64 2_fork
 467 root      20     0   393008 159488 124416 S   3.7    2.0    8:46.69 Xorg
 705 lixuzeng 20     0   3000968 122772 80304 S   3.7    1.5    8:48.10 kwin_x11
19301 lixuzeng 20     0   1844016 117676 35208 S   3.3    1.5    0:25.28 TIM.exe
19304 lixuzeng 20     0    13228   10696   1920 S   3.3    0.1    0:21.77 wineserver
 724 lixuzeng 20     0   1305808 234704 113040 S   2.0    2.9    1:33.80 plasmashell
 6034 lixuzeng 20     0   3266808 542136 188476 S   1.7    6.7    6:29.78 wps
12572 lixuzeng 20     0   472628   71800  58472 S   1.0    0.9    0:31.18 konsole
19380 lixuzeng 20     0   1820460 73164 28452 S   1.0    0.9    0:05.10 TIM.exe
 1021 lixuzeng 20     0   3056536 106600 58308 S   0.7    1.3    0:24.10 sogou-qimpanel
 1033 lixuzeng 20     0    11452   4044   2948 S   0.7    0.1    0:05.75 dbus-daemon
  10 root     -2     0      0      0      0 I   0.3    0.0    0:03.35 rcu_preempt
 610 lixuzeng 20     0    11716   5300   3484 S   0.3    0.1    0:06.60 dbus-daemon
 652 lixuzeng 20     0   315980 39484 34872 S   0.3    0.5    0:03.67 klauncher
 704 lixuzeng 20     0   359068 14732 12088 S   0.3    0.2    0:09.33 mission-control
 722 lixuzeng 20     0   2002068 202884 124716 S   0.3    2.5    0:12.03 krunner
 786 lixuzeng 20     0   480344  87320 71280 S   0.3    1.1    1:07.01 yakuake
1262 lixuzeng 20     0   398704 43220 38304 S   0.3    0.5    0:04.42 akonadi_maildir
1263 lixuzeng 20     0   484924 45592 40280 S   0.3    0.6    0:05.44 akonadi_maildis
1465 lixuzeng 20     0   1406216 299564 150416 S   0.3    3.7    5:55.64 chromium
11167 root     20     0      0      0      0 I   0.3    0.0    0:01.79 kworker/u16:0-events_unbound
14500 root     20     0      0      0      0 I   0.3    0.0    0:01.61 kworker/u16:1-flush-8:0
19322 lixuzeng 20     0   2698384 32028 21496 S   0.3    0.4    0:01.78 QQProtect.exe
  1 root     20     0   201468  9304  7212 S   0.0    0.1    0:01.02 systemd
  2 root     20     0      0      0      0 S   0.0    0.0    0:00.00 kthreadd
  3 root     0 -20     0      0      0 I   0.0    0.0    0:00.00 rcu_gp
  4 root     0 -20     0      0      0 I   0.0    0.0    0:00.00 rcu_par_gp
  6 root     0 -20     0      0      0 I   0.0    0.0    0:00.00 kworker/0:0H-kblockd
  8 root     0 -20     0      0      0 I   0.0    0.0    0:00.00 mm_percpu_wq
  9 root     20     0      0      0      0 S   0.0    0.0    0:00.20 ksoftirqd/0
 11 root     -2     0      0      0      0 I   0.0    0.0    0:00.00 rcu_sched
 12 root     -2     0      0      0      0 I   0.0    0.0    0:00.00 rcu_bh
 13 root     -2     0      0      0      0 S   0.0    0.0    0:00.32 rcuc/0
 14 root     -2     0      0      0      0 S   0.0    0.0    0:00.00 rcub/0
```

利用“ps -aux - sort -pcpu | less”命令根据 cpu 占用率进行降序排序:

```
2: zsh — Konsole
文件(F) 编辑(E) 视图(V) 书签(B) 设置(S) 帮助(H)
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
lixuzeng 17610 99.8  0.0    2296    748 pts/4    R+   15:06   21:59  ./2_fork
lixuzeng  705  5.9  1.5 3001288 122924 ?        Sl   12:58   8:50  /usr/bin/kwin_x11 -session 1014fdca7b1000154296310
100000006450006_1553317033_34917
root      467  5.8  2.0 399692 163012 tty1      Ssl+ 12:58   8:49  /usr/lib/Xorg -nolisten tcp -auth /var/run/sddm/{c
7aca980-906c-445e-b8b2-1926cb37a758} -background none -noreset -displayfd 17 -seat seat0 vt1
lixuzeng  6034  5.2  6.7 3266808 542136 ?        Sl   13:22   6:32  /usr/lib/office6/wps /home/lixuzeng/Downloads/ĖµñĖ
Ėb Ė0³I₂00Ė-2019.docx
lixuzeng 1465  3.9  3.7 1406216 299764 ?        Sl   12:59   5:55  /usr/lib/chromium/chromium
lixuzeng 19301  3.9  1.4 1844016 117676 ?        Sl   15:16   0:27  c:\Program Files\Tencent\TIM\Bin\TIM.exe
lixuzeng 19990  3.6  1.4 561024 113288 ?        Sl   15:27   0:02  /usr/bin/spectacle --dbus
lixuzeng 19304  3.4  0.1 13228 10696 ?        Ss   15:16   0:23  /usr/bin/wineserver
lixuzeng 1493  1.5  1.6 501412 136316 ?        Sl   12:59   2:16  /usr/lib/chromium/chromium --type=gpu-process --fi
eld-trial-handle=10107632335232521299,5848875239985309232,131072 --gpu-preferences=KAAAAAAAAACAAABAAQAAAAAAAAAGAA
AAAAAAAAAAAAAAAAAAAAAgAAAAAAAAAA --service-request-channel-token=11227980969096538283
lixuzeng  9691  1.3  2.2 876492 181148 ?        Sl   13:47   1:18  /usr/lib/chromium/chromium --type=renderer --field
-trial-handle=10107632335232521299,5848875239985309232,131072 --service-pipe-token=15805456150330442173 --lang=zh-C
N --enable-offline-auto-reload --enable-offline-auto-reload-visible-only --num-raster-threads=4 --enable-main-frame
-before-activation --service-request-channel-token=15805456150330442173 --renderer-client-id=80 --no-v8-untrusted-c
ode-mitigations --shared-files=v8_context_snapshot_data:100,v8_natives_data:101
lixuzeng  724  1.0  2.9 1305824 235644 ?        Sl   12:58   1:34  /usr/bin/plasmashell
lixuzeng 19177  1.0  2.0 871788 168308 ?        Sl   14:34   0:32  /usr/lib/chromium/chromium --type=renderer --field
-trial-handle=10107632335232521299,5848875239985309232,131072 --service-pipe-token=10922316018878515277 --lang=zh-C
N --enable-offline-auto-reload --enable-offline-auto-reload-visible-only --num-raster-threads=4 --enable-main-frame
-before-activation --service-request-channel-token=10922316018878515277 --renderer-client-id=102 --no-v8-untrusted-c
ode-mitigations --shared-files=v8_context_snapshot_data:100,v8_natives_data:101
lixuzeng 19141  0.9  1.8 829340 148468 ?        Sl   15:14   0:07  /usr/lib/chromium/chromium --type=renderer --field
-trial-handle=10107632335232521299,5848875239985309232,131072 --service-pipe-token=16096308805001386775 --lang=zh-C
N --enable-offline-auto-reload --enable-offline-auto-reload-visible-only --num-raster-threads=4 --enable-main-frame
-before-activation --service-request-channel-token=16096308805001386775 --renderer-client-id=106 --no-v8-untrusted-c
ode-mitigations --shared-files=v8_context_snapshot_data:100,v8_natives_data:101
lixuzeng 12572  0.8  0.8 472628 71800 ?        Sl   14:27   0:31  /usr/bin/konsole --workdir /home/lixuzeng/05-2019/
实验二
lixuzeng 19380  0.8  0.9 1820460 73164 ?        Sll  15:16   0:05  c:\Program Files\Tencent\TIM\Bin\TIM.exe /hosthwnd
=65690 /hostname=QQ_IPC_{1A230C4F-19B1-44AD-A95A-788E46754C51} /memoryid=0 c:\Program Files\Tencent\TIM\Bin\TIM.exe
lixuzeng  786  0.7  1.0 480344 87320 ?        Sl   12:58   1:07  /usr/bin/yakuake -session 1014fdca7b10001552528317
00000006740033_1553317032_885374
lixuzeng 11533  0.7  2.1 877944 175488 ?        Sl   14:13   0:34  /usr/lib/chromium/chromium --type=renderer --field
-trial-handle=10107632335232521299,5848875239985309232,131072 --service-pipe-token=9875670882903436679 --lang=zh-CN
--enable-offline-auto-reload --enable-offline-auto-reload-visible-only --num-raster-threads=4 --enable-main-frame
-before-activation --service-request-channel-token=9875670882903436679 --renderer-client-id=96 --no-v8-untrusted-cod
e-mitigations --shared-files=v8_context_snapshot_data:100,v8_natives_data:101
lixuzeng 11558  0.7  2.0 864956 166788 ?        Sl   14:14   0:32  /usr/lib/chromium/chromium --type=renderer --field
-trial-handle=10107632335232521299,5848875239985309232,131072 --service-pipe-token=470140078417399384 --lang=zh-CN
--enable-offline-auto-reload --enable-offline-auto-reload-visible-only --num-raster-threads=4 --enable-main-frame-b
efore-activation --service-request-channel-token=470140078417399384 --renderer-client-id=97 --no-v8-untrusted-code-
```

从得到的结果可以看出父进程在进行不断循环 cpu 占用率高达 99.8%。

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。



程序代码：

```
#include<unistd.h>
#include<stdio.h>
int main()
{
    pid_t fpid = fork();
```


一般来说，在 `fork()` 之后是父进程先执行还是子进程先执行是不确定的。这取决于内核所使用的调度算法。（然而实验多次并为出现次序改变）

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。

程序代码：

```
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
int main()
{
    pid_t fpid = fork();
    if(fpid == 0)
    {
        while(1){
            printf("我是子进程 P3 ， 我的 pid 为： %d,我的 ppid 为： %d\n",getpid(),getppid());
            sleep(2);
        }
    }
    else
    {
        fpid = fork();
        if(fpid == 0)
        {
            fpid = fork();
            if(fpid == 0)
            {
                while(1){
                    printf("我是孙子进程 P4 ， 我的 pid 为： %d, 我的 ppid 为： %d\n",getpid(),getppid());
                    sleep(2);
                }
            }
            else
            {
                fpid = fork();
                if(fpid == 0)
                {
                    while(1){
                        printf("我是孙子进程 P5 ， 我的 pid 为： %d,ppid 为： %d\n",getpid(),getppid());
                        sleep(2);
                    }
                }
            }
        }
    }
}
```



```

    }
    else
    {

    }
}
while(1){
    printf("我是子进程 P2 , 我的 pid 为: %d,我的 ppid 为: %d\n",getpid(),getppid());
    sleep(2);
    int *ptr = (int *)0;
    *ptr = 100;
}
}
else
{

}
while(1){
    printf("我是父进程 P1 , 我的 pid 为: %d,我的 ppid 为: %d\n",getpid(),getppid());
    sleep(2);
}
}
sleep(20);
return 0;
}

```

运行结果:

1>使用 kill -9 进程号命令将 P2 进程终止:

The left terminal window shows the output of the program. It displays a series of messages from P1 and P2 processes, including their PID and PPID. The messages are repeated, indicating a loop in the execution.

The right terminal window shows the use of the `pstree` command to visualize the process tree. It shows P1 as the parent process, with P2 as its child. The `kill -9 14422` command is used to terminate P2. The `ps -c 14422` command is used to check the status of P2, which is shown as `z` (zombie). The `ps -c 14422` command is also used to check the status of P1, which is shown as `T` (running).

P1 正常运行, 从 ps 命令可以看出 P2 进程变为僵尸进程 (stat 状态为 z, zombie), 其子进程 P4,P5 仍在运行, 但是变为 pid 为 1 的子进程, 相对于 P1 的进程树不显示 P4,P5。

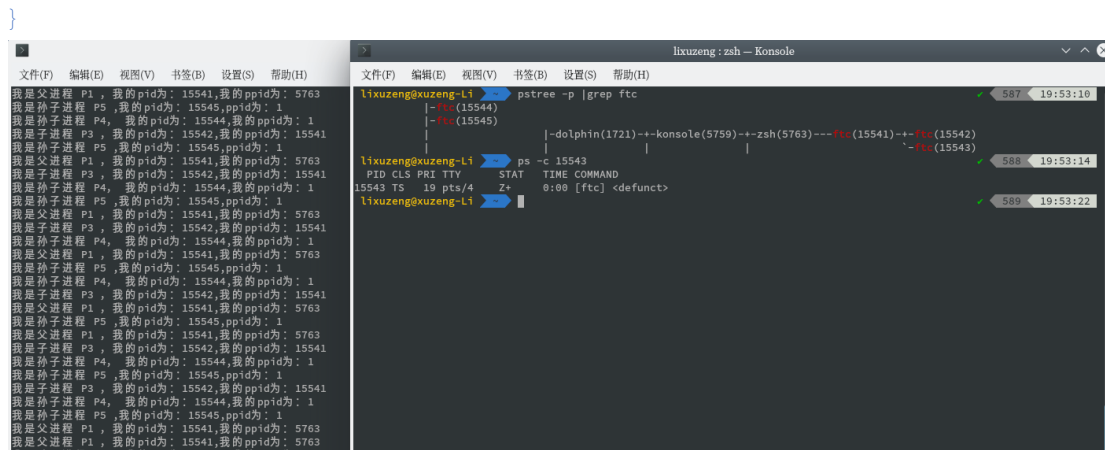
2>使用 exit() 使进程 P2 正常退出:

修改代码:

```

while(1){
    printf("我是子进程 P2 , 我的 pid 为: %d,我的 ppid 为: %d\n",getpid(),getppid());
    sleep(2);
    exit(1)
}

```



P1 正常运行,同使用 kill 命令杀死进程相同 P2 进程变为僵尸进程(stat 状态为 z,zombie), 其子进程 P4,P5 仍在运行,但是变为 pid 为 1 的子进程,相对于 P1 的进程树不显示 P4,P5。

3>采用段错误方式退出

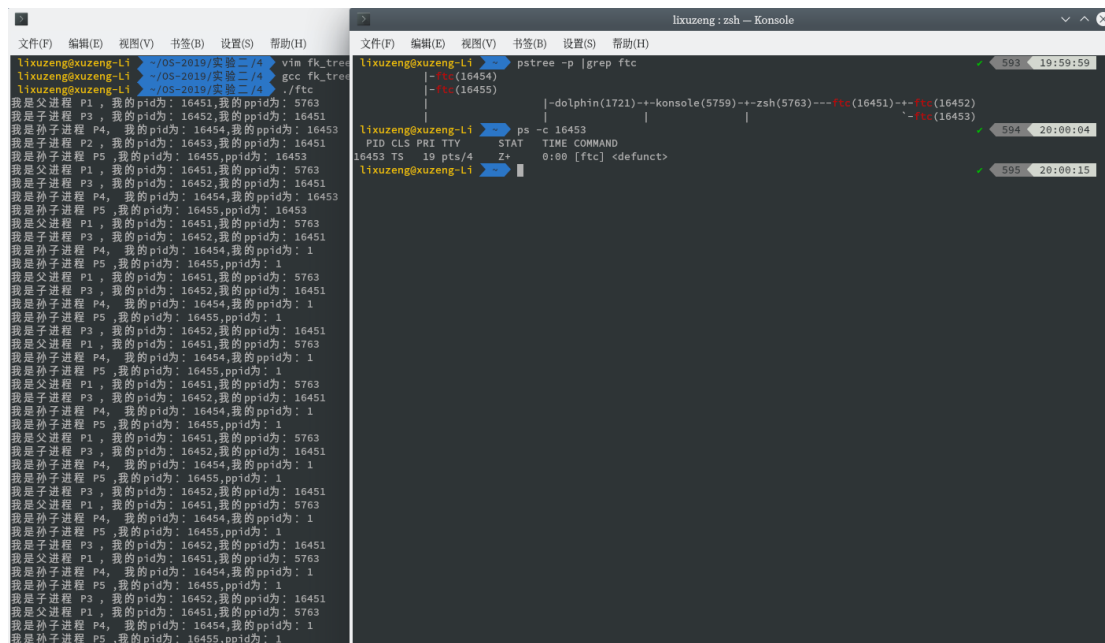
段错误产生原因:

- 1、访问不存在的内存地址;
- 2、访问系统保护的内存地址
- 3、访问只读的内存地址
- 4、栈溢出

这里我采用访问系统保护的地址使段错误产生:

```
while(1){  
    printf("我是子进程 P2 , 我的 pid 为: %d,我的 ppid 为: %d\n",getpid(),getppid());  
    sleep(2);  
    int *ptr = (int *)0;  
    *ptr = 100;  
}
```

运行结果:



P1 正常运行, 同使用前两种方式结束 P2 进程相同, P2 进程变为僵尸进程 (stat 状态为 z,

zombie), 其子进程 P4, P5 仍在运行但是变为 pid 为 1 的子进程, 相对于 P1 的进程树不显示 P4, P5。