



上海工程技术大学

实 验 报 告

实验三 进程管理

1. 实验要求

- (1) 掌握操作系统中进程的基本概念;
- (2) 掌握 Linux 操作系统进程管理基本方法;
- (3) 了解进程并发运行的过程。

2. 实验内容

1) 验证实验:

- ① 启动 vi 编辑器, 并将其调入后台, 再调出前台, 最后退出 vi;
- ② 启动 vi 编辑器, 并将其调入后台, 再将该作业杀死;
- ③ 启动两个 vi 编辑器, 均调入后台后, 查看当前有那些作业正在执行;
- ④ 列出你在当前系统中的所有进程。指出你在该系统中最早启动的进程是那个, 在什么时候启动?
- ⑤ 列出当前系统中占用 CPU 时间比例最高的进程;
- ⑥ 系统内核的进程中那个进程的运行时间最长, 为什么?
- ⑦ 先后新建文件 a 和文件 b, 并相应启动 vi 对它们进行编辑。一开始要求文件 a 在前台执行, b 在后台执行。对文件 a 输入一些文字后, 将其切换到后台, 然后将文件 b 调到前台, 同样输入一些文字。最后将打开 a 文件的 vi 进程杀死, b 文件保存并退出 vi;
- ⑧ 列出当前系统中的所有进程(包括普通进程和守护进程), 并指出哪些进程的优先级较高, 这些进程有什么特点?
- ⑨ 守护进程中那些进程优先级比较高, 那些进程的优先级比较低, 你能发现优先级比较低的是些什么样的进程? 优先级比较高的又是什么样的进程? 请找一两个典型的进程加以说明;
- ⑩ 利用 nice 程序启动三个 vi 程序, 其 nice 分别为 5, 10, 15, 观察这三个 vi 程序的优先级, nice 值与进程的优先级呈怎样的数值关系? 或者说, nice 值每增加 1, 优先级的数值是否也增加 1, 还是增加得更多? (可利用 renice 命令不断调整进程的 nice 值以获得实际规律)。

- 2) 编程实验, 在 C 语言程序中嵌入使用进程管理、同步与通信相关系统调用, 观察程序运行过程并分析程序输出结果:

- ① 创建子进程，让父进程和子进程分别打印 10 次“This is parent/son process, process id is XX”；
- ② 父子进程分别从终端读入字符串并打印该命令字符串；
- ③ 在题目②的基础上，让子进程打印后退出，父进程等待子进程结束并打印子进程的返回值；
- ④ 编写程序。要求在程序中使用系统调用 `lockf()` 来给每一个进程加锁，实现进程之间的互斥，观察并分析出现的现象；
- ⑤ 编一段程序，使其实现进程的软中断通信。

要求使用系统调用 `fork()` 创建两个子进程，再用系统调用 `signal()` 让父进程捕捉键盘上来的中断信号；当捕捉到中断信号后，父进程用系统调用 `kill()` 向两个子进程发出信号，子进程捕捉到信号后分别输出下列信息后终止：

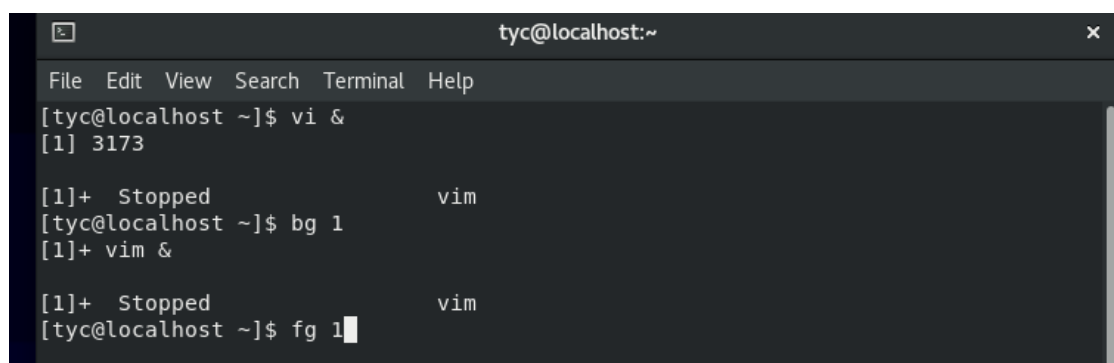
Child process 1 is killed by parent!

Child process 2 is killed by parent!

父进程等待两个子进程终止后，输出如下的信息后终止：Parent process is killed!

- ⑥ 新建一个程序。要求程序实现功能：在主函数中首先设置信号 `SIGINT`（可由按键“Ctrl+C”产生）的处理方式，处理方式为执行一段用户预定义的信号处理函数 `int_func()`。当进程截获从控制终端键入的“Ctrl+C”后转入执行 `int_func()` 函数，再次设置信号 `SIGINT` 的处理方式为缺省处理方式（终止进程）并显示信号值。观察程序运行过程并分析程序输出结果；
- ⑦ 补充选做题：用信号量实现生产者消费者问题的进程的同步控制。

3. 实验结果



```
tyc@localhost:~  
File Edit View Search Terminal Help  
[tyc@localhost ~]$ vi &  
[1] 3173  
  
[1]+ Stopped vim  
[tyc@localhost ~]$ bg 1  
[1]+ vim &  
  
[1]+ Stopped vim  
[tyc@localhost ~]$ fg 1
```

```
tyc@localhost:~  
File Edit View Search Terminal Help  
[tyc@localhost ~]$ vi &  
[1] 3252  
  
[1]+ Stopped vim  
[tyc@localhost ~]$ bg 1  
[1]+ vim &  
  
[1]+ Stopped vim  
[tyc@localhost ~]$ kill 3252  
[tyc@localhost ~]$
```

```
[tyc@localhost ~]$ jobs -l  
[tyc@localhost ~]$ vi &  
[1] 3344  
  
[1]+ Stopped vim  
[tyc@localhost ~]$ jobs -l  
[1]+ 3344 Stopped (tty output) vim  
[tyc@localhost ~]$ vi&  
[2] 3359  
  
[2]+ Stopped vim  
[tyc@localhost ~]$ bg 1  
[1]- vim &  
  
[1]+ Stopped vim  
[tyc@localhost ~]$ jobs -l  
[1]+ 3344 Stopped (tty output) vim  
[2]- 3359 Stopped (tty output) vim  
[tyc@localhost ~]$
```

```
[tyc@localhost ~]$ ps -aux  
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND  
root         1  0.5  0.3 252428 2940 ?        Ss   03:18   0:03 /usr/lib/syst  
root         2  0.0  0.0      0     0 ?        S    03:18   0:00 [kthreadd]  
root         3  0.0  0.0      0     0 ?        I<   03:18   0:00 [rcu_gp]  
root         4  0.0  0.0      0     0 ?        I<   03:18   0:00 [rcu_par_gp]  
root         5  0.0  0.0      0     0 ?        I    03:18   0:00 [kworker/0:0-  
root         6  0.0  0.0      0     0 ?        I<   03:18   0:00 [kworker/0:0H  
root         7  0.0  0.0      0     0 ?        I    03:18   0:00 [kworker/0:1-  
root         8  0.0  0.0      0     0 ?        I    03:18   0:00 [kworker/u256  
root         9  0.0  0.0      0     0 ?        I<   03:18   0:00 [mm_percpu_wq  
root        10  0.0  0.0      0     0 ?        S    03:18   0:00 [ksoftirqd/0]  
root        11  0.0  0.0      0     0 ?        R    03:18   0:00 [rcu_sched]  
root        12  0.0  0.0      0     0 ?        S    03:18   0:00 [migration/0]  
root        13  0.0  0.0      0     0 ?        S    03:18   0:00 [watchdog/0]  
root        14  0.0  0.0      0     0 ?        S    03:18   0:00 [cpuhp/0]  
root        16  0.0  0.0      0     0 ?        S    03:18   0:00 [kdevtmpfs]  
root        17  0.0  0.0      0     0 ?        I<   03:18   0:00 [netns]  
root        18  0.0  0.0      0     0 ?        S    03:18   0:00 [kauditd]  
root        19  0.0  0.0      0     0 ?        S    03:18   0:00 [khungtaskd]  
root        20  0.0  0.0      0     0 ?        S    03:18   0:00 [oom_reaper]  
root        21  0.0  0.0      0     0 ?        I<   03:18   0:00 [writeback]  
root        22  0.0  0.0      0     0 ?        S    03:18   0:00 [kcompactd0]  
root        23  0.0  0.0      0     0 ?        SN   03:18   0:00 [ksmd]  
root        24  0.0  0.0      0     0 ?        SN   03:18   0:00 [khugepaged]  
root        25  0.0  0.0      0     0 ?        I<   03:18   0:00 [crypto]  
root        26  0.0  0.0      0     0 ?        I<   03:18   0:00 [kintegrityd]  
root        27  0.0  0.0      0     0 ?        I<   03:18   0:00 [kblockd]  
root        28  0.0  0.0      0     0 ?        I<   03:18   0:00 [blkcg_punt_b  
root        29  0.0  0.0      0     0 ?        I<   03:18   0:00 [tpm_dev_wq]  
root        30  0.0  0.0      0     0 ?        I<   03:18   0:00 [md]  
root        31  0.0  0.0      0     0 ?        I<   03:18   0:00 [edac-poller]  
root        32  0.0  0.0      0     0 ?        S    03:18   0:00 [watchdogd]  
root        33  0.2  0.0      0     0 ?        I<   03:18   0:01 [kworker/0:1H  
root        35  0.1  0.0      0     0 ?        R    03:18   0:00 [kworker/u256  
root        55  1.8  0.0      0     0 ?        S    03:18   0:09 [kswapd0]
```

```
tyc@localhost:~$ ps -aux --sort rss
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	2	0.0	0.0	0	0 ?		S	03:18	0:00	[kthreadd]
root	3	0.0	0.0	0	0 ?		I<	03:18	0:00	[rcu_gp]
root	4	0.0	0.0	0	0 ?		I<	03:18	0:00	[rcu_par_gp]
root	5	0.0	0.0	0	0 ?		I	03:18	0:00	[kworker/0:0-
root	6	0.0	0.0	0	0 ?		I<	03:18	0:00	[kworker/0:0H
root	7	0.0	0.0	0	0 ?		I	03:18	0:00	[kworker/0:1-
root	8	0.0	0.0	0	0 ?		I	03:18	0:00	[kworker/u256
root	9	0.0	0.0	0	0 ?		I<	03:18	0:00	[mm_percpu_wq
root	10	0.0	0.0	0	0 ?		S	03:18	0:00	[ksoftirqd/0]
root	11	0.0	0.0	0	0 ?		I	03:18	0:00	[rcu_sched]
root	12	0.0	0.0	0	0 ?		S	03:18	0:00	[migration/0]
root	13	0.0	0.0	0	0 ?		S	03:18	0:00	[watchdog/0]
root	14	0.0	0.0	0	0 ?		S	03:18	0:00	[cpuhp/0]
root	16	0.0	0.0	0	0 ?		S	03:18	0:00	[kdevtmpfs]
root	17	0.0	0.0	0	0 ?		I<	03:18	0:00	[netns]
root	18	0.0	0.0	0	0 ?		S	03:18	0:00	[kauditd]
root	19	0.0	0.0	0	0 ?		S	03:18	0:00	[khungtaskd]
root	20	0.0	0.0	0	0 ?		S	03:18	0:00	[oom_reaper]
root	21	0.0	0.0	0	0 ?		I<	03:18	0:00	[writeback]
root	22	0.0	0.0	0	0 ?		S	03:18	0:00	[kcompactd0]
root	23	0.0	0.0	0	0 ?		SN	03:18	0:00	[ksmd]
root	24	0.0	0.0	0	0 ?		SN	03:18	0:00	[khugepaged]
root	25	0.0	0.0	0	0 ?		I<	03:18	0:00	[crypto]
root	26	0.0	0.0	0	0 ?		I<	03:18	0:00	[kintegrityd]
root	27	0.0	0.0	0	0 ?		I<	03:18	0:00	[kblockd]
root	28	0.0	0.0	0	0 ?		I<	03:18	0:00	[blkcg_punt_b
root	29	0.0	0.0	0	0 ?		I<	03:18	0:00	[tpm_dev_wq]
root	30	0.0	0.0	0	0 ?		I<	03:18	0:00	[md]
root	31	0.0	0.0	0	0 ?		I<	03:18	0:00	[edac-poller]
root	32	0.0	0.0	0	0 ?		S	03:18	0:00	[watchdogd]
root	33	0.2	0.0	0	0 ?		I<	03:18	0:01	[kworker/0:1H
root	35	0.0	0.0	0	0 ?		I	03:18	0:00	[kworker/u256
root	55	1.6	0.0	0	0 ?		S	03:18	0:09	[kswapd0]

```
tyc@localhost:~$ ps -aux --sort rss
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.5	0.3	252428	2916 ?		Ss	03:18	0:03	/usr/lib/syst
tyc	2512	0.0	0.3	849224	2940 ?		Ssl	03:20	0:00	/usr/libexec/
root	928	0.0	0.3	553836	2952 ?		Ssl	03:18	0:00	/usr/libexec/
tyc	2242	0.0	0.3	85496	3044 ?		Ssl	03:19	0:00	/usr/bin/dbus
tyc	2390	0.0	0.3	836272	3120 tty2		Sl	03:19	0:00	/usr/libexec/
tyc	2476	0.0	0.3	621704	3144 tty2		Sl+	03:20	0:00	/usr/libexec/
tyc	2683	0.0	0.3	622972	3144 tty2		SNl+	03:20	0:00	/usr/libexec/
tyc	2565	0.0	0.4	523112	3236 tty2		Sl+	03:20	0:00	/usr/libexec/
root	2819	0.0	0.4	425828	3452 ?		Sl	03:20	0:00	/usr/sbin/abr
tyc	2207	0.0	0.4	108956	3576 ?		Ss	03:19	0:00	/usr/lib/syst
tyc	2506	0.0	0.4	510084	3624 tty2		Sl+	03:20	0:00	/usr/libexec/
dbus	925	0.2	0.4	73864	3640 ?		Ssl	03:18	0:01	/usr/bin/dbus
root	612	0.0	0.4	98064	3708 ?		Ss	03:18	0:00	/usr/lib/syst
root	1116	0.0	0.4	474996	3896 ?		Ssl	03:19	0:00	/usr/sbin/Net
tyc	2377	0.0	0.4	389028	3896 tty2		Sl	03:19	0:00	ibus-daemon -
tyc	2717	0.0	0.5	605400	4032 tty2		Sl+	03:20	0:00	abrt-applet
root	1026	0.0	0.5	229132	4076 ?		S	03:19	0:00	/usr/libexec/
tyc	2547	0.0	0.5	606844	4112 tty2		Sl+	03:20	0:00	/usr/libexec/
root	2854	0.0	0.5	641864	4288 ?		Ssl	03:20	0:00	/usr/libexec/
tyc	2358	0.0	0.5	562180	4364 tty2		Sl+	03:19	0:00	/usr/bin/Xway
tyc	3508	0.0	0.6	61996	5436 pts/0		R+	03:28	0:00	ps -aux --sor
tyc	3472	0.2	0.7	28440	5696 pts/0		Ss	03:28	0:00	bash
tyc	2716	0.0	0.7	548344	5856 ?		Ssl	03:20	0:00	/usr/libexec/
tyc	2556	0.0	0.7	1120904	5956 tty2		Sl+	03:20	0:00	/usr/libexec/
polkitd	931	0.6	0.7	1637612	6112 ?		Ssl	03:18	0:03	/usr/lib/polkit
tyc	2674	0.0	0.7	640684	6180 tty2		SNl+	03:20	0:00	/usr/libexec/
tyc	2386	0.1	0.7	436776	6344 tty2		Sl	03:19	0:00	/usr/libexec/
tyc	2227	0.0	1.1	1247712	9164 ?		S-sl	03:19	0:00	/usr/bin/puls
tyc	2675	0.1	1.9	537528	15536 tty2		Sl+	03:20	0:00	/usr/bin/vmto
root	2470	0.1	2.8	219620	22712 ?		Ss	03:20	0:00	/usr/libexec/
tyc	3467	2.2	4.7	518700	38192 ?		Rsl	03:28	0:00	/usr/libexec/
tyc	2720	0.3	5.3	1002488	42772 tty2		Sl+	03:20	0:01	/usr/bin/gnom
root	2024	1.9	10.6	769500	85184 ?		Ssl	03:19	0:10	/usr/libexec/
tyc	2330	3.3	16.5	2968836	132376 tty2		Sl+	03:19	0:17	/usr/bin/gnom


```

#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/wait.h>

int main(int arg,char* argv[]){
    // int 被 typedef 为 pid_t
    pid_t pid=fork();

    // 当pid==0时,是子进程代码运行区域。其他则是父进程运行区域。
    if(pid<0){
        printf("Create child process failure ...\n");

    }else if(pid==0){
        //子进程执行体
        char str[100]={0};
        printf("请输入字符串:\t");
        gets(str);
        printf("输入的字符串是: \t");
        puts(str);
    }
    else{
        //父进程执行体
        char str[100]={0};
        printf("请输入字符串:\t");
        gets(str);
        printf("输入的字符串是: \t");
    }
}

```

-- 插入 --

1,19

顶端

```

        printf("输入的字符串是: \t");
        puts(str);
    }

    // 执行体结束标志
    if(pid==0)
    {
        printf("pid=%i child process end ... \n",getpid());
    }
    else{
        // 睡眠5s,等待子先进程结束
        sleep(5);
        int status=0;
        wait(status);
        int returned = WEXITSTATUS(status);
        printf("exited normally with status %d\n",returned);
        printf("pid=%i Parent process End ... \n",getpid());
    }

    return 0;
}

```

-- 插入 --

47,2


```

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
ltd@ltd-virtual-machine:~$ gcc string.c -o string
string.c: In function 'main':
string.c:18:9: warning: implicit declaration of function 'gets'; did you mean '
fgets'? [-Wimplicit-function-declaration]
    gets(str);
    ^~~~~
    fgets
string.c:40:14: warning: passing argument 1 of 'wait' makes pointer from intege
r without a cast [-Wint-conversion]
    wait(status);
    ^~~~~~
In file included from string.c:4:0:
/usr/include/x86_64-linux-gnu/sys/wait.h:77:16: note: expected 'int *' but argu
ment is of type 'int'
    extern __pid_t wait (int *__stat_loc);
                  ^~~~~
/tmp/cc5Lc00L.o: 在函数'main'中:
string.c:(.text+0x87): 警告: the `gets' function is dangerous and should not b
e used.
ltd@ltd-virtual-machine:~$ ./string
请输入字符串: 请输入字符串: ljxshltd
输入的字符串是:      ljxshltd

输入的字符串是:
pid=19098 child process end ...
exited normally with status 0
pid=19097 Parent process End ...

```

```

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
void int_func (int sig)

{printf (" in int_func, receive signal =%d \n", sig );

    signal(SIGINT,SIG_DFL );    // 改变为缺省处理方式（终止进程！）

    return;
}
void main()

{ signal(SIGINT, int_func );    // 函数调用设置信号处理方式

    while(1)

{printf (" Hello world! \n" );

    sleep(1);        // 入睡1 秒

}

}

```

```

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
ldd@ldd-virtual-machine:~$ vi sg.c
ldd@ldd-virtual-machine:~$ gcc sg.c -o sg
ldd@ldd-virtual-machine:~$ ./sg
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
^C in int_func, receive signal =2
Hello world!
Hello world!
Hello world!
^C
ldd@ldd-virtual-machine:~$

```

```

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```

```

#include<stdio.h>
#include <signal.h>
#include <unistd.h>
int main() {
    pid_t p1, p2;
    p1 = fork();
    if(p1<0) puts("error");
    else if(p1==0)
    {
        lockf(1,1,0);
        puts("子进程 1");
        lockf(1,0,0);
    }
    else
    {
        p2=fork();
        if(p2<0) puts("error");
        else if (p2==0)
        {
            lockf(1,1,0);
            puts("子进程 2");
            lockf(1,0,0);
        }
        else
        {
            lockf(1,1,0);
            puts("父进程");

```

```

        lockf(1,0,0);
    }
}

return 0;
}

```



```
ldd@ldd-virtual-machine:~$ vi lock.c
ldd@ldd-virtual-machine:~$ gcc lock.c -o lock
ldd@ldd-virtual-machine:~$ ./lock
父进程
子进程 2
子进程 1
^C
ldd@ldd-virtual-machine:~$
```

```
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>
#include <sys/wait.h>
int waiting_control=1;
void waiting();
void sig(int num);
int main() {
    pid_t p1, p2;
    p1 = fork();
    if(p1<0) puts("error");
    else if(p1==0)
    {
        lockf(1,1,0);
        puts("子进程 1");
        lockf(1,0,0);
        signal(SIGINT,sig);
        waiting();
        printf("Child process 1 is killed by parent\n");
    }
    else
    {
        p2=fork();
        if(p2<0) puts("error");
        else if (p2==0)
        {
```

```

    lockf(1,1,0);
    puts("子进程 2");
    lockf(1,0,0);
    signal(SIGINT,sig);
    waiting();
    printf("Child process 2 is killed by parent\n");
}
else
{
    lockf(1,1,0);
    puts("父进程");
    lockf(1,0,0);
    signal(SIGINT,sig);
    waiting();
    kill(p1,SIGINT);
    kill(p2,SIGINT);
    wait(0);
    wait(0);
    printf("Parent process is killed\n");
}
}
return 0;

```

```

    return 0;
}
void sig(int num)
{
    waiting_control=0;
}

```

```

void waiting()
{
    while(waiting_control==1);
}

```

```
ldd@ldd-virtual-machine:~$ vi signal.c
ldd@ldd-virtual-machine:~$ gcc signal.c -o signal
ldd@ldd-virtual-machine:~$ ./signal
父进程
子进程 2
子进程 1
^Cchild process 1 is killed by parent
Child process 2 is killed by parent
Parent process is killed
ldd@ldd-virtual-machine:~$
```

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
#include <stdio.h>
#include<stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <sys/wait.h>
void waiting( );
void stop( );
int wait_mark;
int main( )
{
    int p1,p2,stdout;
    while((p1=fork( ))!=-1);        /* 创建子进程 p1*/
    if (p1>0)
    {
        while((p2=fork( ))!=-1);    /* 创建子进程 p2*/
        if (p2>0)
        {
            wait_mark=1;
            signal(SIGINT,stop);    /* 接收到 ^c 信号, 转 stop*/
            waiting( );
            kill(p1,16);            /* 向 p1 发软中断信号 16*/
            kill(p2,17);            /* 向 p2 发软中断信号 17*/
            wait(0);                /* 同步 */
            wait(0);
            printf("Parent process is killed!\n");
            exit(0);
        }
    }
}
```

```
    else
    {
        wait_mark=1;
        signal(17,stop);    /* 接收到软中断信号 17 , 转 stop*/
        waiting( );
        lockf(1,1,0);
        printf("Child process 2 is killed by parent!\n");
        lockf(1,0,0);
        exit(0);
    }
}
else
{
    wait_mark=1;
    signal(16,stop);        /* 接收到软中断信号 16 , 转 stop*/
    waiting( );
    lockf(1,1,0);
    printf("Child process 1 is killed by parent!\n");
    lockf(1,0,0);
    exit(0);
}
}
void waiting( )
{
    while(wait_mark!=0);
}
void stop( )
```

```
void stop( )  
{  
wait_mark=0;  
}
```