实验三 守护进程

**一、实验目的**

1、了解守护进程的生命周期及应用。

2、掌握编写守护进程的五个基本步骤。

**二、实验原理**

1、守护进程是运行在后台的一种特殊进程，用于周期性地执行某种任务或等待处理某些发生的事件。

2、守护进程生存期较长，常常在系统引导装入时启动，在系统关闭时终止。守护进程的名称通常以d结尾。

3、编写守护进程的五个步骤：

（1）fork()创建子进程，exit()退出父进程。

（2）setsid()在子进程中创建新会话，使子进程脱离原来的控制。

（3）chdir()改变当前的工作目录为根目录，防止可卸载的文件系统被占用。

（4）umask()设置文件权限掩码，防止继承来的文件创建屏蔽字，拒绝某些权限。

（5）close()关闭文件描述符，由于子进程从父进程继承一些打开的无用的文件，

需要将其关闭，防止占用资源

**三、实验内容**

1、编写守护进程test，test每5秒钟打印一个数字，定向输出到trush.txt。

2、编写并编译monitor.c，其功能为每5秒检测一次test是否正在运行；若未运行，则运行该程序。

3、先验证test是否能正常运行，需要执行test，然后用命令查看数字是否正常输出至trush.txt。

4、然后执行kill命令终止进程，使用命令查看test此时并未运行。

5、执行monitor，5秒钟后使用命令查看test此时已经运行。

**选做部分：**将编写的守护进程设置为开机自启动。

## 四, 程序源码

1.

daemon.h

#ifndef \_LAB3\_DAEMON\_H

#define \_LAB3\_DEAMON\_H

void to\_be\_daemon();

void write\_log(const char \*proc, char \*message);

#endif

daemon.c

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<unistd.h>

#include<signal.h>

#include<syslog.h>

void to\_be\_daemon(){

pid\_t pid;

if((pid = fork()) < 0){

printf("can't fork!\n");

exit(1);

}else if(pid > 0){

exit(0);

}

if(setsid() < 0){

printf("setsid error\n");

exit(1);

}

umask(0);

chdir("/");

for(int i = 0; i < getdtablesize(); i++){

close(i);

}

}

void write\_log(const char \*proc, char \*message){

openlog(proc, LOG\_PID|LOG\_CONS, LOG\_USER);

syslog(LOG\_INFO, "%s", message);

closelog();

}

2. 守护进程 test

#include<stdio.h>

#include<unistd.h>

#include<fcntl.h>

#include"daemon.h"

const char \*proc = "daemon-test";

int main(int argc, char \*argv[]){

to\_be\_daemon();

write\_log(proc, "daemon-test begin running!\n");

while(1){

sleep(5);

int fd = open("/home/lambdafate/trush.txt", O\_CREAT | O\_WRONLY | O\_APPEND, 0777);

if(fd == -1){

write\_log(proc, "fd == -1\n");

}else{

write(fd, "hello\n", 6);

}

close(fd);

}

return 0;

}

3 . 守护进程 monitor

#include<stdio.h>

#include<unistd.h>

#include"daemon.h"

int check\_proc();

void exec\_proc();

const char \*test\_path = "/home/lambdafate/projects/embedded-os/lab3-daemon/";

const char \*proc = "daemon-monitor";

int main(int argc, char \*argv[]){

to\_be\_daemon();

write\_log(proc, "daemon-monitor begin running!\n");

while(1){

sleep(5);

if(check\_proc() == -1){

write\_log(proc, "test process no running? don't worry, i will run it!\n");

exec\_proc();

}

}

return 0;

}

// check "./test" process status

int check\_proc(){

const char \*cmd = "ps -aux | grep ./test | grep -v grep";

char buffer[32];

FILE \*fp = NULL;

if((fp = popen(cmd, "r")) == NULL){

return -1;

}

int res = 1;

if(fgets(buffer, sizeof(buffer), fp) == NULL){

res = -1;

}

pclose(fp);

return res;

}

// using fork-exec to execute test

void exec\_proc(){

if(fork() == 0){

write\_log(proc, "exec\_proc() -> fork success!\n");

// change work dir

chdir(test\_path);

// child process

if(execl("./test", "./test", NULL) == -1){

write\_log(proc, "exec\_proc() -> execl error!\n");

}

}

}

### 五. 测试

1. 直接运行 test

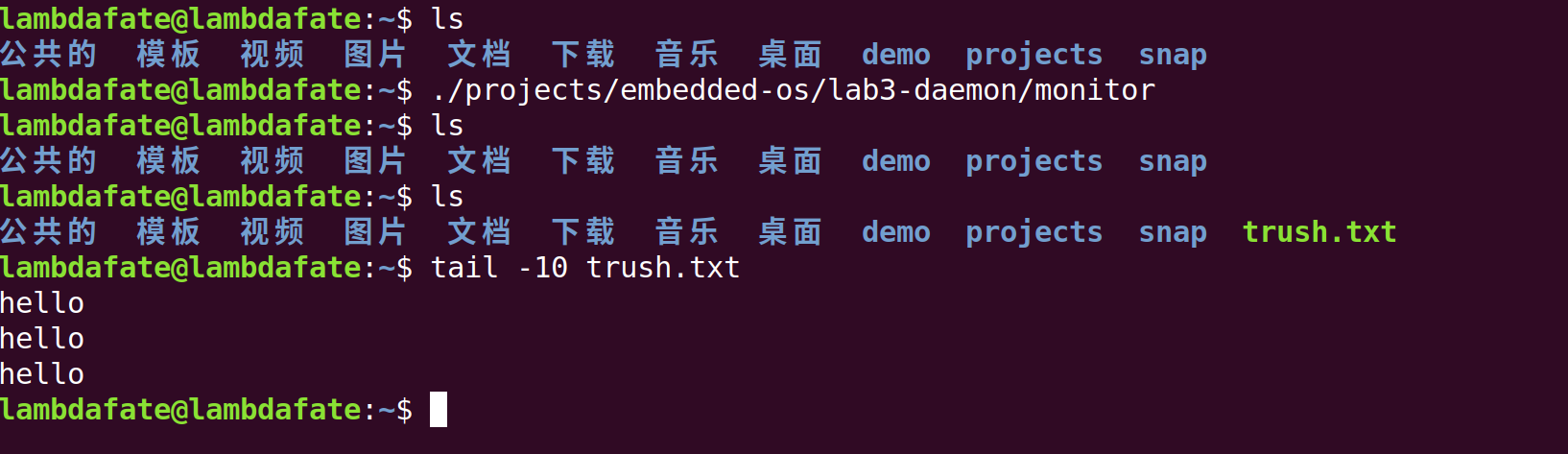


2. 将monitor 编写为守护进程， 检查test进程是否存在，如果不存在则启动test进程

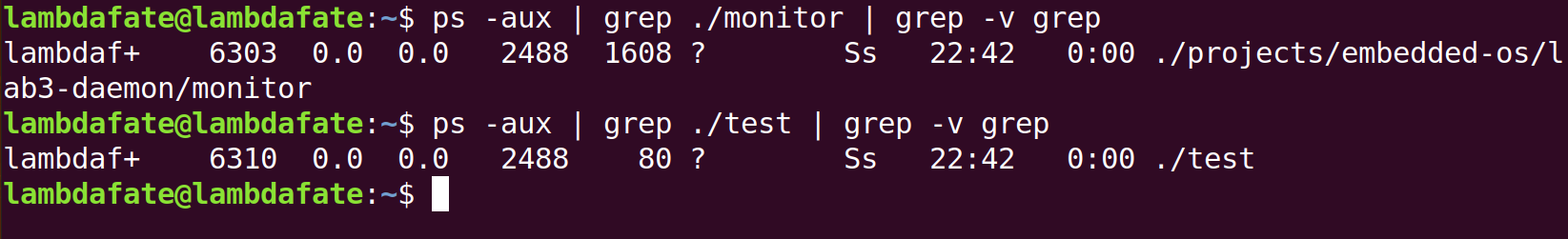
首先kill掉test进程，并删除test的输出文件



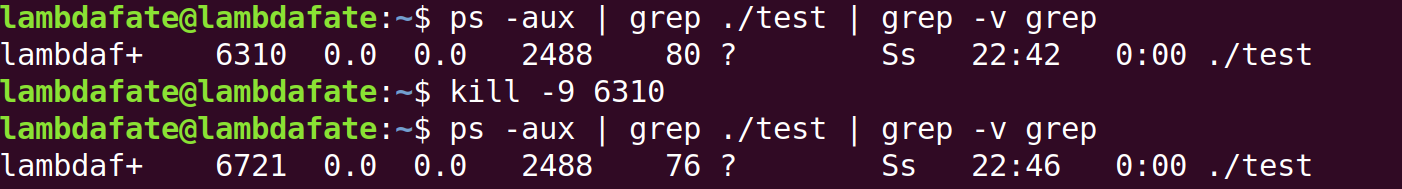
然后启动 monitor 进程



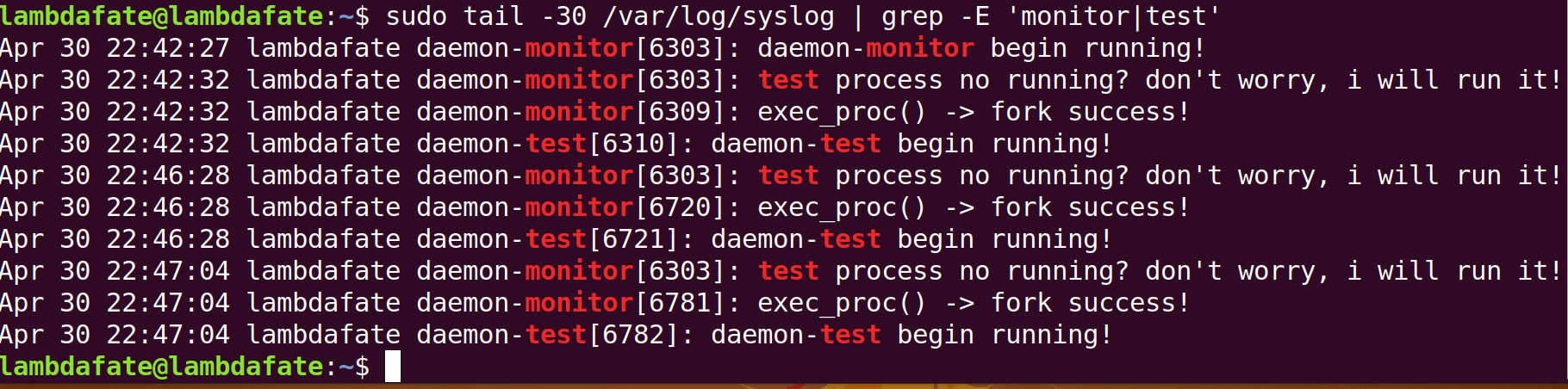
查看test进程和monitor进程



Kill掉 test进程 (可以看到 test进程又被monitor重启)



查看log记录(和上面的kill操作对应)



**选做部分：**将编写的守护进程设置为开机自启动

只要将monitor设置为开机启动即可

1. 在terminal 中输入 crontab -e
2. 在最后添加启动脚本 ：

@reboot /home/lambdafate/projects/embedded-os/lab3-daemon/monitor

如图：

