同济大学计算机系

计算机网络课程实验报告

学　　号 **1552239**

姓　　名 岳昊玮

专　　业 计算机科学与技术

授课老师 沈坚

一、守护进程的编写(进阶)以及 systemctl 的使用、rpm 安装包的生成

1. 每个人的目录结构要求如下（假设学号为 1551234，各人按实修改）：首先建立"学号-000106"子

   目录（可位于任意子目录下），下层不需要再建子目录，示例如下：

   1551234-000106

   |-- 本次作业的各文件，含 makefile

2. 写一个满足以下各要求的 test 程序

2.1 运行后成为进程，但是不要完全脱离控制台，即在启动控制台上用 ps 仍能查到，

只需要把之前代码中的 setsid()注释掉就好了。

```
[root@Anokoro 1552239-000106]# ps
  PID TTY          TIME CMD
 2263 pts/0    00:00:00 bash
 2341 pts/0    00:00:05 test
 2342 pts/0    00:00:00 ps
```

2.2 修改进程名，使用 ps 命令查看时，在父进程的后面加[main]，随后每隔一秒，分裂出 1 个子进程，共 n 个（5≤n≤100），每个子进程在后面加[sub-xx]

对于进程名的修改，如果使用 prctl（PR_SET_NAME, new_name）的话，如下图中，可以看到，我使用 ps 命令查看的时候，进程名已经被修改了，但是上面那个框中，我使用 ps -aux 查询的时候，进程名还是原来的，对照进程号可以看出这两个是同一个进程。此时进程名称并没有改变，改变的只是/prco/ (PID)/status 和/proc/(PID)/stat 的值，而/prco/$(PID)/cmdline 并没有改变（上面那个 test 是上面那道题的，运行之后没有 kill 掉）

```
root       2374 54.6  0.0   6324    96 pts/0    R    15:03   0:04 ./test
root       2376  0.0  0.3 153120  1780 pts/0    R+   15:03   0:00 ps -aux
[root@Anokoro 1552239-000106]# ps
  PID TTY          TIME CMD
 2263 pts/0    00:00:00 bash
 2341 pts/0    00:10:53 test
 2374 pts/0    00:00:10 ./test[main]
 2377 pts/0    00:00:00 ps
[root@Anokoro 1552239-000106]#
```

通过修改进程 argv[0]修改进程名，只需要在进程启动时修改 argv[0]所指向的内存空间的内容，就可以修改进程名，在修改之前先来查看一下 argv[0]的值

添加代码如下：

```
printf("\nargv[%d]:%s\n",i,argv[i]);
```

打印输出结果如下

```
[root@Anokoro 1552239-000106]# ./test
[root@Anokoro 1552239-000106]#
argv[0]:./test
```

可以看到 argv[0]的确是我们的进程名，接下来开始修改
在 main 中添加调用如下两个函数（具体的定义见 test.c）

```
//修改argv[0]所指向的内存空间的内容
setproctitle_init(argc, argv, environ);

 //调用prctl修改进程名
setproctitle("%s%s", "./test", "[main]");
```

```
root       2494 90.3  0.0   6456   324 pts/0    R    15:37   0:05 ./test[main]
root       2496  0.0  0.3 153120  1780 pts/0    R+   15:37   0:00 ps -aux
[root@Anokoro 1552239-000106]# ps
  PID TTY          TIME CMD
 2263 pts/0    00:00:00 bash
 2494 pts/0    00:00:07 ./test[main]
 2497 pts/0    00:00:00 ps
[root@Anokoro 1552239-000106]#
```

然后分裂子进程，子进程代码如下：

```
void sub(){
        int pid;
        pid = fork();
        if(pid > 0){
                prctl(PR_SET_NAME,"./test [main]");
        }
        else if (pid < 0){
                printf("子进程分裂失败\n");
        }
        else if(pid == 0 ){

                char mm[20] = "./test [sub-";
                char mmm[4];
                itoa(i,mmm,10);
                strcat(mm,mmm);
                strcat(mm,"]\0");

                prctl(PR_SET_NAME,mm);

                while(1);
                }

        return ;
}
```

在 main 函数里面的调用

```
while(1)
{
        sleep(1);
        sub();
        i++;
}
```

执行结果

```
[root@Anokoro 1552239-000106]# ps
   PID TTY          TIME CMD
  2168 pts/0    00:00:00 bash
  2211 pts/0    00:00:00 ./test [main]
  2212 pts/0    00:00:02 ./test [sub-1]
  2214 pts/0    00:00:01 ./test [sub-2]
  2215 pts/0    00:00:01 ./test [sub-3]
  2218 pts/0    00:00:00 ./test [sub-4]
  2221 pts/0    00:00:00 ./test [sub-5]
  2225 pts/0    00:00:00 ./test [sub-6]
  2228 pts/0    00:00:00 ./test [sub-7]
  2231 pts/0    00:00:00 ./test [sub-8]
```

增加代码

```
if(pid > 0){
        strcpy(argv[0],"./test [main]");
        prctl(PR_SET_NAME,"./test [main]");
}
        else if (pid < 0){
        printf("子进程分裂失败\n");
}
else if(pid == 0 ){

        char mm[20] = "./test [sub-";
        char mmm[4];
        itoa(i,mmm,10);
        strcat(mm,mmm);
        strcat(mm,"]\0");

        prctl(PR_SET_NAME,mm);
        strcpy(argv[0],mm);
        while(1);
}
```

使用 ps -ef 查看

```
[root@Anokoro 1552239-000106]# ps -ef |grep test
root       2234      1  0 12:10 pts/0    00:00:00 ./test [main]
root       2236   2234 26 12:10 pts/0    00:00:02 ./test [sub-1]
root       2238   2234 19 12:10 pts/0    00:00:01 ./test [sub-2]
root       2240   2234 16 12:10 pts/0    00:00:01 ./test [sub-3]
root       2242   2234 14 12:10 pts/0    00:00:01 ./test [sub-4]
root       2245   2234 13 12:10 pts/0    00:00:00 ./test [sub-5]
root       2247   2234 11 12:10 pts/0    00:00:00 ./test [sub-6]
root       2248   2234 11 12:10 pts/0    00:00:00 ./test [sub-7]
root       2249   2234 10 12:10 pts/0    00:00:00 ./test [sub-8]
root       2250   2234  9 12:10 pts/0    00:00:00 ./test [sub-9]
root       2251   2234  8 12:10 pts/0    00:00:00 ./test [sub-10]
root       2252   2234  6 12:10 pts/0    00:00:00 ./test [sub-11]
root       2254   2182  0 12:10 pts/0    00:00:00 grep --color=auto test
```

2.3 修改进程名，使用 ps -ef 命令查看时，父进程和子进程的名称的后面再加上自己的已运行时间（以秒为单位），即父进程分裂 n 个子进程完成后只要子进程不退出，只需要每秒更新一次自己的运行时间即可，不需要做其它操作；子进程运行后，只需要每秒更新一次自己的运行时间，不需要做其它操作

main 的程序运行时间：在程序开始运行时添加

```
time_t time_start;
time_t run_time;
time_start = time(NULL);
char time_buf[20];
```

在每次循环中增加

```
run_time = time(NULL) - time_start;

strftime(time_buf,20,"%H:%M:%S]",gmtime(&run_time));
```

sub 的程序运行时间：在子程序代码段的运行时间计算

```
else if(pid == 0 ){
    signal(SIGHUP, handle_signal);
    prctl(PR_SET_PDEATHSIG, SIGHUP); //在子进程中设置当父进程退出
    time_t start_sub = time(NULL);
    while(1){
        time_t run_time_sub = time(NULL) - start_sub;
        strftime(time_buf,20," %H:%M:%S]",gmtime(&run_time_sub));
```

```
[root@Anokoro 1552239-000106]# ps -ef |grep test
root       2317      1  0 14:08 pts/0    00:00:00 ./test [main 00:00:21]
root       2318   2317 17 14:08 pts/0    00:00:03 ./test [sub-1 00:00:20]
root       2319   2317 13 14:08 pts/0    00:00:02 ./test [sub-2 00:00:19]
root       2321   2317 11 14:08 pts/0    00:00:02 ./test [sub-3 00:00:18]
root       2322   2317 10 14:08 pts/0    00:00:01 ./test [sub-4 00:00:17]
root       2323   2317  9 14:08 pts/0    00:00:01 ./test [sub-5 00:00:16]
root       2324   2317  8 14:08 pts/0    00:00:01 ./test [sub-6 00:00:15]
root       2325   2317  7 14:08 pts/0    00:00:01 ./test [sub-7 00:00:14]
root       2326   2317  7 14:08 pts/0    00:00:01 ./test [sub-8 00:00:13]
root       2327   2317  6 14:08 pts/0    00:00:00 ./test [sub-9 00:00:12]
root       2328   2317  6 14:08 pts/0    00:00:00 ./test [sub-10 00:00:11]
root       2329   2317  6 14:08 pts/0    00:00:00 ./test [sub-11 00:00:10]
root       2330   2317  6 14:08 pts/0    00:00:00 ./test [sub-12 00:00:09]
root       2331   2317  5 14:08 pts/0    00:00:00 ./test [sub-13 00:00:08]
root       2332   2317  5 14:08 pts/0    00:00:00 ./test [sub-14 00:00:07]
root       2333   2317  5 14:08 pts/0    00:00:00 ./test [sub-15 00:00:06]
root       2334   2317  5 14:08 pts/0    00:00:00 ./test [sub-16 00:00:05]
```

2.4 用 kill 杀掉若干子进程后，父进程会再次分裂，补齐 n 个，且子进程的编号占用之前被 kill 子进程的编号

代码修改如下

```
void sub_killed()
{
      int ret;
      while((ret = waitpid(-1,NULL,WNOHANG)) > 0){
              subNum--;
              int a =0;
              for(;pid_sub[a] > 0;a++)
                      if (pid_sub[a] == ret) {
                      i = a +1 ;
                      break;
                      }
      }
}
```

子进程退出的信号处理

```
int fileRead();

int  main(int argc , char *argv[])
  {
      daemon(1,1);
      signal(SIGCHLD,sub_killed);
```

```
if(subNum < n){
        subNum++;
        i=subNum+1;
}
```

这是两个计数变量之间的关系

运行效果

```
[root@Anokoro 1552239-000106]# ps -ef |grep test
root        8647      1  0 18:10 ?        00:00:00 ./test [main 00:00:10]
root        8648   8647 31 18:10 ?        00:00:02 ./test [sub-1 00:00:10]
root        8649   8647 23 18:10 ?        00:00:01 ./test [sub-2 00:00:09]
root        8650   8647 19 18:10 ?        00:00:01 ./test [sub-3 00:00:08]
root        8651   8647 17 18:10 ?        00:00:01 ./test [sub-4 00:00:07]
root        8652   8647 15 18:10 ?        00:00:00 ./test [sub-5 00:00:06]
root        8655   8647 14 18:10 ?        00:00:00 ./test [sub-6 00:00:05]
root        8656   8647 13 18:10 ?        00:00:00 ./test [sub-7 00:00:04]
root        8659   8647 13 18:10 ?        00:00:00 ./test [sub-8 00:00:03]
root        8662   8647 14 18:10 ?        00:00:00 ./test [sub-9 00:00:02]
root        8665   8647  0 18:10 ?        00:00:00 ./test [sub-10 00:00:01]
root        8667   2466  0 18:10 pts/1    00:00:00 grep --color=auto test
[root@Anokoro 1552239-000106]# kill 8655 8662 8649

[root@Anokoro 1552239-000106]# ps -ef |grep test
root        8647      1  0 18:10 ?        00:00:00 ./test [main 00:00:31]
root        8648   8647 16 18:10 ?        00:00:04 ./test [sub-1 00:00:31]
root        8650   8647 11 18:10 ?        00:00:03 ./test [sub-3 00:00:29]
root        8651   8647 11 18:10 ?        00:00:03 ./test [sub-4 00:00:28]
root        8652   8647 10 18:10 ?        00:00:02 ./test [sub-5 00:00:27]
root        8656   8647 10 18:10 ?        00:00:02 ./test [sub-7 00:00:25]
root        8659   8647  9 18:10 ?        00:00:02 ./test [sub-8 00:00:24]
root        8665   8647  9 18:10 ?        00:00:02 ./test [sub-10 00:00:22]
root        8668   8647  9 18:10 ?        00:00:00 ./test [sub-2 00:00:10]
root        8669   8647  9 18:10 ?        00:00:00 ./test [sub-6 00:00:10]
root        8670   8647  9 18:10 ?        00:00:00 ./test [sub-9 00:00:10]
root        8672   2466  0 18:10 pts/1    00:00:00 grep --color=auto test
```

2.5 Kill 父进程后，所有子进程自动结束（注意：不允许杀子进程）

父进程退出信号处理函数

```
void handle_signal(int signo)
{
    if (signo == SIGHUP){
        //printf("child exit...\n");
        exit(0);
      }
}
```

子进程中增加

```
else if(pid == 0 ){
    signal(SIGHUP, handle_signal);
    prctl(PR_SET_PDEATHSIG, SIGHUP);
```

杀掉父进程，子进程自动退出

```
[root@Anokoro 1552239-000106]# ps -ef | grep test
root       2249      1  0 14:58 pts/0    00:00:00 ./test [main 00:00:13]
root       2250   2249 26 14:58 pts/0    00:00:03 ./test [sub-1 00:00:13]
root       2251   2249 19 14:58 pts/0    00:00:02 ./test [sub-2 00:00:12]
root       2253   2249 16 14:58 pts/0    00:00:01 ./test [sub-3 00:00:11]
root       2254   2249 14 14:58 pts/0    00:00:01 ./test [sub-4 00:00:10]
root       2255   2249 13 14:58 pts/0    00:00:01 ./test [sub-5 00:00:09]
root       2256   2249 12 14:58 pts/0    00:00:00 ./test [sub-6 00:00:08]
root       2257   2249 11 14:58 pts/0    00:00:00 ./test [sub-7 00:00:07]
root       2258   2249 10 14:58 pts/0    00:00:00 ./test [sub-8 00:00:06]
root       2259   2249 10 14:58 pts/0    00:00:00 ./test [sub-9 00:00:05]
root       2260   2249 10 14:58 pts/0    00:00:00 ./test [sub-10 00:00:04]
root       2261   2249 10 14:58 pts/0    00:00:00 ./test [sub-11 00:00:03]
root       2262   2249 11 14:58 pts/0    00:00:00 ./test [sub-12 00:00:02]
root       2263   2249  0 14:58 pts/0    00:00:00 ./test [sub-13 00:00:01]
root       2265   2195  0 14:58 pts/0    00:00:00 grep --color=auto test
[root@Anokoro 1552239-000106]#
[root@Anokoro 1552239-000106]#
[root@Anokoro 1552239-000106]# kill -9 2249
[root@Anokoro 1552239-000106]# ps -ef | grep test
root       2274   2195  0 14:58 pts/0    00:00:00 grep --color=auto test
[root@Anokoro 1552239-000106]#
```

2.6 n 值从 /etc 目录下的 1551234.conf 文件中读取，文件中仅一行，存放运行时需要分裂子进程的数量，合理范围 5-100，如果超过合理范围，缺省为 5 增加如下代码

```
int n,subNum = 0;

FILE *file=fopen("/etc/1552239.conf","r")
while(fgetc(file) != '=');
fscanf(file,"%d",&n);
fclose(file);
if(n < 5 || n > 100) n = 5;
```

并且给 fork()函数的调用增加条件。

```
if(subNum < n)
        pid = fork();
else pid = 1;
if(pid > 0){
```

运行效果如下（以 10 个子进程为例）通过查看运行的时间可知当分裂了 10 个子进程之后便不再继续分裂。

```
[root@Anokoro 1552239-000106]# ps -ef | grep test
root       4050      1  0 10:13 pts/1    00:00:00 ./test [main 00:00:16]
root       4051   4050 22 10:13 pts/1    00:00:03 ./test [sub-1 00:00:15]
root       4052   4050 17 10:13 pts/1    00:00:02 ./test [sub-2 00:00:14]
root       4053   4050 14 10:13 pts/1    00:00:01 ./test [sub-3 00:00:13]
root       4054   4050 13 10:13 pts/1    00:00:01 ./test [sub-4 00:00:12]
root       4055   4050 12 10:13 pts/1    00:00:01 ./test [sub-5 00:00:11]
root       4058   4050 11 10:13 pts/1    00:00:01 ./test [sub-6 00:00:10]
root       4059   4050 10 10:13 pts/1    00:00:00 ./test [sub-7 00:00:09]
root       4062   4050 10 10:13 pts/1    00:00:00 ./test [sub-8 00:00:08]
root       4063   4050  9 10:13 pts/1    00:00:00 ./test [sub-9 00:00:07]
root       4066   4050  9 10:14 pts/1    00:00:00 ./test [sub-10 00:00:06]
root       4076   2466  0 10:14 pts/1    00:00:00 grep --color=auto test
[root@Anokoro 1552239-000106]#
```

2.7. 读配置文件的函数编译为动态链接库，命名为 lib1551234.so，放入 /usr/lib64 中（注意：test 程序打开的是 /usr/lib64/lib1551234.so，不是本目录下的）

```
[root@Anokoro 1552239-000106]# make clean
rm -f *.o *.so test  /usr/lib64/lib1552239.so
[root@Anokoro 1552239-000106]#
[root@Anokoro 1552239-000106]# pkill test
[root@Anokoro 1552239-000106]#
[root@Anokoro 1552239-000106]#
[root@Anokoro 1552239-000106]# make
gcc -c -fPIC 1552239.c
gcc -o   lib1552239.so  -shared 1552239.o
gcc test.c  -L. -l1552239 -o test
cp ./lib1552239.so /usr/lib64
[root@Anokoro 1552239-000106]#
[root@Anokoro 1552239-000106]# ./test
[root@Anokoro 1552239-000106]#
```

clean 时也删掉/usr/lib64下的.so

cp一份.so到/usr/lib

运行时无需添加路径，会自动去默认/usr/lib64下寻找.so

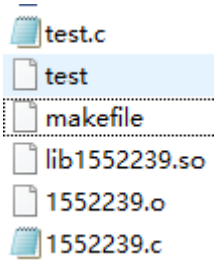
2.8. 在/usr 下新建 1551234 子目录，里面放一个 1551234.dat 文件，内容为学号+姓名，与可执行程序无关，可执行程序不需要读写此文件

```
[root@Anokoro 1552239-000106]# cat /usr/1552239/1552239.dat
1552239岳昊玮
```

2.9 makefile 文件

make ：生成可执行文件、动态链接库文件

```
[root@Anokoro 1552239-000106]# make
gcc -c -fPIC 1552239.c
gcc -o   lib1552239.so  -shared 1552239.o
gcc test.c  -L.  -l1552239 -o test
[root@Anokoro 1552239-000106]#
```



make install：将可执行文件及附属文件放入指定位置

-f 删除已经存在的目标文件而不提示。

-r 若给出的源文件是一目录文件，此时 cp 将递归复制该目录下所有的子目录和文件。此时目标文件必须为一个目录名

```
[root@Anokoro 1552239-000106]# make install
cp -rf ./test /usr/sbin/test-1552239
cp -rf ./lib1552239.so  /usr/lib64
mkdir -p /usr/1552239 && cp -rf ./1552239.dat  "/usr/1552239"
cp -rf ./1552239.conf /etc
[root@Anokoro 1552239-000106]#
```

make rpm ：生成 rpm 安装包

makefile 内容

```
rpm:
	mkdir /usr/1552239/rpmbuild/{BUILD,BUILDROOT,SOURCES,SPECS,RPMS,SRPMS} -p
	mkdir test-1552239
	cp *.*  makefile   ./test-1552239
	tar -czf test-1552239.tar.gz  ./test-1552239
	rm  -rf  ./test-1552239
	cp test-1552239.tar.gz /usr/1552239/rpmbuild/SOURCES
	cp ./test-1552239.spec  /usr/1552239/rpmbuild/SPECS
	mkdir  /usr/1552239/rpmbuild/BUILDROOT/test-1552239-1.x86_64
	ln -s /usr   /usr/1552239/rpmbuild/BUILDROOT/test-1552239-1.x86_64/usr
	ln -s /etc   /usr/1552239/rpmbuild/BUILDROOT/test-1552239-1.x86_64/etc
	rpmbuild -ba /usr/1552239/rpmbuild/SPECS/test-1552239.spec
	cp /usr/1552239/rpmbuild/RPMS/x86_64/test-1552239-1.x86_64.rpm  ./test-1552239.rpm
	cp /usr/1552239/rpmbuild/SRPMS/test-1552239-1.src.rpm ./test-1552239.src.rpm
```

执行结果

```
[root@Anokoro 1552239-000106]# make rpm
mkdir /usr/1552239/rpmbuild/{BUILD,BUILDROOT,SOURCES,SPECS,RPMS,SRPMS} -p
mkdir test-1552239
cp *.*  makefile   ./test-1552239
tar -czf test-1552239.tar.gz  ./test-1552239
rm  -rf  ./test-1552239
cp test-1552239.tar.gz /usr/1552239/rpmbuild/SOURCES
cp ./test-1552239.spec  /usr/1552239/rpmbuild/SPECS
mkdir  /usr/1552239/rpmbuild/BUILDROOT/test-1552239-1.x86_64
ln -s /usr   /usr/1552239/rpmbuild/BUILDROOT/test-1552239-1.x86_64/usr
ln -s /etc   /usr/1552239/rpmbuild/BUILDROOT/test-1552239-1.x86_64/etc
rpmbuild -ba /usr/1552239/rpmbuild/SPECS/test-1552239.spec
执行(%prep): /bin/sh -e /var/tmp/rpm-tmp.JcGmlI
+ umask 022
+ cd /usr/1552239/rpmbuild/BUILD
+ cd /usr/1552239/rpmbuild/BUILD
+ rm -rf test-1552239
+ /usr/bin/gzip -dc /usr/1552239/rpmbuild/SOURCES/test-1552239.tar.gz
+ /usr/bin/tar -xf -
+ STATUS=0
+ '[' 0 -ne 0 ']'
+ cd test-1552239
+ /usr/bin/chmod -Rf a+rX,u+w,g-w,o-w .
+ exit 0
执行(%build): /bin/sh -e /var/tmp/rpm-tmp.aU6215
+ umask 022
+ cd /usr/1552239/rpmbuild/BUILD
+ cd test-1552239
+ make
make[1]: Entering directory `/usr/1552239/rpmbuild/BUILD/test-1552239'
gcc test.c  -L.  -l1552239 -o test
make[1]: Leaving directory `/usr/1552239/rpmbuild/BUILD/test-1552239'
+ exit 0
处理文件：test-1552239-1.x86_64
Provides: config(test) = 1552239-1 lib1552239.so()(64bit) test = 1552239-1
Requires(interp): /bin/sh /bin/sh /bin/sh /bin/sh
Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1 rpmlib(FileDigest
Requires(pre): /bin/sh
Requires(post): /bin/sh
Requires(preun): /bin/sh
Requires(postun): /bin/sh
Requires: lib1552239.so()(64bit) libc.so.6()(64bit) libc.so.6(GLIBC_2.2.5)(64bit) libc.so
检查未打包文件：/usr/lib/rpm/check-files /usr/1552239/rpmbuild/BUILDROOT/test-1552239-1.x
写道:/usr/1552239/rpmbuild/SRPMS/test-1552239-1.src.rpm
写道:/usr/1552239/rpmbuild/RPMS/x86_64/test-1552239-1.x86_64.rpm
执行(%clean): /bin/sh -e /var/tmp/rpm-tmp.1gMR1f
+ umask 022
+ cd /usr/1552239/rpmbuild/BUILD
+ cd test-1552239
+ rm -rf /usr/1552239/rpmbuild/BUILDROOT/test-1552239-1.x86_64
+ make clean
make[1]: Entering directory `/usr/1552239/rpmbuild/BUILD/test-1552239'
rm -f *.o *.so test  /usr/lib64/lib1552239.so *.rpm
make[1]: Leaving directory `/usr/1552239/rpmbuild/BUILD/test-1552239'
+ exit 0
cp /usr/1552239/rpmbuild/RPMS/x86_64/test-1552239-1.x86_64.rpm  ./test-1552239.rpm
cp /usr/1552239/rpmbuild/SRPMS/test-1552239-1.src.rpm ./test-1552239.src.rpm
[root@Anokoro 1552239-000106]#
```

make uninstall：清除指定位置的可执行文件及附属文件

makefile 内容

```
uninstall:
        rm -f  /usr/sbin/test-1552239
        rm -f /usr/lib64/$(LIB)
        rm -f /etc/1552239.conf /usr/1552239/1552239.dat
        rm -rf /usr/1552239/rpmbuild
        rm -f /etc/systemd/system/test-1552239.service
        -rmdir   /usr/1552239
```

执行结果

```
[root@Anokoro 1552239-000106]# make uninstall
rm -f  /usr/sbin/test-1552239
rm -f /usr/lib64/lib1552239.so
rm -f /etc/1552239.conf /usr/1552239/1552239.dat
rm -rf /usr/1552239/rpmbuild
rm -f /etc/systemd/system/test-1552239.service
rmdir   /usr/1552239
```

make clean：清除本目录下的文件

makefile 内容

```
clean:
        rm -f *.o *.so $(target)  /usr/lib64/$(LIB) *.rpm
```

执行结果

```
[root@Anokoro 1552239-000106]# make clean
rm -f *.o *.so test  /usr/lib64/lib1552239.so *.rpm
```

3. 写一个 test-1551234.service ，放在特定目录下，能完成如下要求：

systemctl enable test-1551234.service ： 使开机时自动运行 test-1551234 进程
systemctl disable test-1551234.service ： 取消开机时自动运行 test-1551234 进程
systemctl start test-1551234.sevice ： 运行 test-1551234 进程
systemctl stop test-1551234.service ： 停止运行 test-1551234 进程
systemctl restart test-1551234.service ： 先停止，再次运行 test-1551234 进程

似乎是在 7 之前 service 控制脚本必须放在 /etc/init.d/ 目录下，而且这个文件的编写非常繁琐，在 7 之后变得非常简单

```
/etc/systemd/system/
```

将 test-1552239.service 放在这个目录下面

test-1552239.service 内容如下

```
[Unit]
Description= test

[Service]
ExecStart = /usr/sbin/test-1552239
ExecStop = /usr/sbin/kill test-1552239
ExecStatus = /usr/sbin/test-1552239 status
Restart  = on-failure
```

通知 systemd 有个新服务添加
systemctl daemon-reload

```
[root@Anokoro ~]# systemctl start  test-1552239
[root@Anokoro ~]# systemctl restart  test-1552239
[root@Anokoro ~]#
[root@Anokoro ~]#
[root@Anokoro ~]# pstree
systemd-+-./test [main]---9*[./test-1552239 ]
        |-NetworkManager---2*[{NetworkManager}]
        |-VGAuthService
        |-abrt-watch-log
        |-abrtd
        |-atd


[root@Anokoro ~]# systemctl stop  test-1552239
[root@Anokoro ~]# pstree
systemd-+-NetworkManager---2*[{NetworkManager}]
        |-VGAuthService
        |-abrt-watch-log
```

4. 将 test 程序及附件共同打包成一个 rpm 安装包

首先安装 rpm-build 打包软件

```
yum install rpmbuild
|# yum install rpmdevtools
```

制作 rpm 包的过程：

BUILD 源代码解压后的存放目录
BUILDROOT 软件 make install 的测试安装目录，也就是测试中的根
RPMS 制作完成后的 RPM 包存放目录，里面有与平台相关的子目录
SOURCES 收集的源材料，补丁的存放位置
SPECS SPEC 文件存放目录
SRMPS 存放 SRMPS 生成的目录

Step1： 在用户主目录下创建一个 RPM 构建根目录结构，也就是生成工作空间

运行

```
# rpmdev-setuptree
```

进入工作目录，查看文件

```
[root@Anokoro rpmbuild]# ls
BUILD  BUILDROOT  RPMS  SOURCES  SPECS  SRPMS
```

Step2：把源码压缩成 tar.gz 放到 SOURCES 下
Step3：在 SPECS 下建立 1552239.spec 文件

Step4： 用 rpmbuild 命令制作 rpm 包，rpmbuild 命令会根据 spec 生成 rpm 包

```
[root@Anokoro rpmbuild]# rpmbuild -ba SPECS/1552239.spec
```

制作 rpm 最主要的是 **spec（说明书）的编写**

软件包基本参数：

```
Name:       test
Version:    1552239
Release:    1
Summary:    test
License:    GPL
Group:      Applications/Archiving
Source: %{name}-%{version}.tar.gz
BuildRoot:      %{_tmppath}/%{name}-%{version}-root

%description
This is for test.
```

自定义打包参数：

预处理段，解压源码包放在 BUILD 目录下面，-q 不显示解压信息

```
%prep
%setup -q
```

编译安装段，，执行 make 命令

```
%build
make
```

安装阶段

```
%pre
if [$1 == 0]; then
    echo "准备安装 test-1552239"
fi
%insitall
#先删除原来安装的
rm -rf %{buildroot}
make install
#$1==1 代表是第一次安装 ，2代表是升级 0代表卸载
%post
    echo "完成安装   test-1552239..."

%preun
if [ $1 == 0 ];then
    echo "准备卸载 test-1552239..."
fi
make uninstall
%postun
    echo "完成卸载 test-1552239..."
```

完成 build 之后报如下错误

```
处理文件：test-1552239-1.x86_64
错误：没有找到文件：/root/rpmbuild/BUILDROOT/test-1552239-1.x86_64/usr/sbin/test-1552239
错误：没有找到文件：/root/rpmbuild/BUILDROOT/test-1552239-1.x86_64/usr/lib64/lib1552239.so
错误：没有找到文件：/root/rpmbuild/BUILDROOT/test-1552239-1.x86_64/usr/1552239/1552239.dat
错误：没有找到文件：/root/rpmbuild/BUILDROOT/test-1552239-1.x86_64/etc/1552239.conf


RPM 构建错误：
    没有找到文件：/root/rpmbuild/BUILDROOT/test-1552239-1.x86_64/usr/sbin/test-1552239
    没有找到文件：/root/rpmbuild/BUILDROOT/test-1552239-1.x86_64/usr/lib64/lib1552239.so
    没有找到文件：/root/rpmbuild/BUILDROOT/test-1552239-1.x86_64/usr/1552239/1552239.dat
    没有找到文件：/root/rpmbuild/BUILDROOT/test-1552239-1.x86_64/etc/1552239.conf
```

建立两个软连接，指向资源文件、配置文件所在的/etc 和 /usr 目录

```
[root@Anokoro rpmbuild]# ln -s  /etc  /root/rpmbuild/BUILDROOT/test-1552239-1.x86_64/etc
[root@Anokoro rpmbuild]# ln -s  /usr  /root/rpmbuild/BUILDROOT/test-1552239-1.x86_64/usr
```

好了，重新打包，然后又是下一个坑 T^T

```
检查未打包文件：/usr/lib/rpm/check-fi
错误：发现已安装(但未打包的)文件：
   /etc
   /usr
```

这个倒比较好解决

在文件的%file 后面（基本上也就是文件的最后一行了）， 加上 /* 重新运行就好了，

还有一个问题是 rpm 的构建的工作路径的问题，这个默认的工作路径是在~/.rpmmacros 文件里面的%_topdir 设置的，我把我的路径设置成了我自己的路径，怕是会出问题哦。T^T

下面是制作 rpm 运行过程的截图，使用-ba 参数制作出*.rpm 和 src.rpm

```
[root@Anokoro rpmbuildJ# rpmbuild -ba SPECS/test-1552239.spec
执行(%prep): /bin/sh -e /var/tmp/rpm-tmp.HNxYOK
+ umask 022
+ cd /root/rpmbuild/BUILD
+ cd /root/rpmbuild/BUILD
+ rm -rf test-1552239
+ /usr/bin/tar -xf -
+ /usr/bin/bzip2 -dc /root/rpmbuild/SOURCES/test-1552239.tar.gz
+ STATUS=0
+ '[' 0 -ne 0 ']'
+ cd test-1552239
+ /usr/bin/chmod -Rf a+rX,u+w,g-w,o-w .
+ exit 0
执行(%build): /bin/sh -e /var/tmp/rpm-tmp.sLnm8Q
+ umask 022
+ cd /root/rpmbuild/BUILD
+ cd test-1552239
+ make
gcc -c -fPIC 1552239.c
gcc -o   lib1552239.so  -shared 1552239.o
gcc test.c  -L.  -l1552239 -o test
+ exit 0
处理文件：test-1552239-1.x86_64
Provides: config(test) = 1552239-1 lib1552239.so()(64bit) test = 1552239-
Requires(interp): /bin/sh /bin/sh /bin/sh /bin/sh
Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1 rpmlib(FileDiges
Requires(pre): /bin/sh
Requires(post): /bin/sh
Requires(preun): /bin/sh
Requires(postun): /bin/sh
Requires: lib1552239.so()(64bit) libc.so.6()(64bit) libc.so.6(GLIBC_2.2.5
检查未打包文件：/usr/lib/rpm/check-files /root/rpmbuild/BUILDROOT/test-15
写道:/root/rpmbuild/SRPMS/test-1552239-1.src.rpm
写道:/root/rpmbuild/RPMS/x86_64/test-1552239-1.x86_64.rpm
执行(%clean): /bin/sh -e /var/tmp/rpm-tmp.LrCz4a
+ umask 022
+ cd /root/rpmbuild/BUILD
+ cd test-1552239
+ rm -rf /root/rpmbuild/BUILDROOT/test-1552239-1.x86_64
+ make clean
rm -f *.o *.so test  /usr/lib64/lib1552239.so *.rpm
+ exit 0
```

4.3. 用 rpm -e 卸载后，要依次删除上面的 5 个文件及 1 个目录，但如果 /usr/1551234 目录下还有其他文件，则仅删除 1551234.dat 文件而保留目录

这个要先把 1552239.dat 删除掉，然后再删目录，删目录的时候如果目录时空的就删除，如果不是空的就不删除，在 makefile 中添加

```
rmdir -rf /usr/1552239
```