

守护进程的编写 systemctl和rpm

1 test运行后成为进程，但是不完全脱离控制台

fork两次并且不setid(), 就可以让程序成为后台进程而不脱离控制台

手动成为守护进程的函数:

```
void daemon_by_hand()
{
    pid_t pid;
    pid = fork();
    if (pid == 0)
    {
    }
    else
        exit(0);
    //skip setsid
    //setsid();

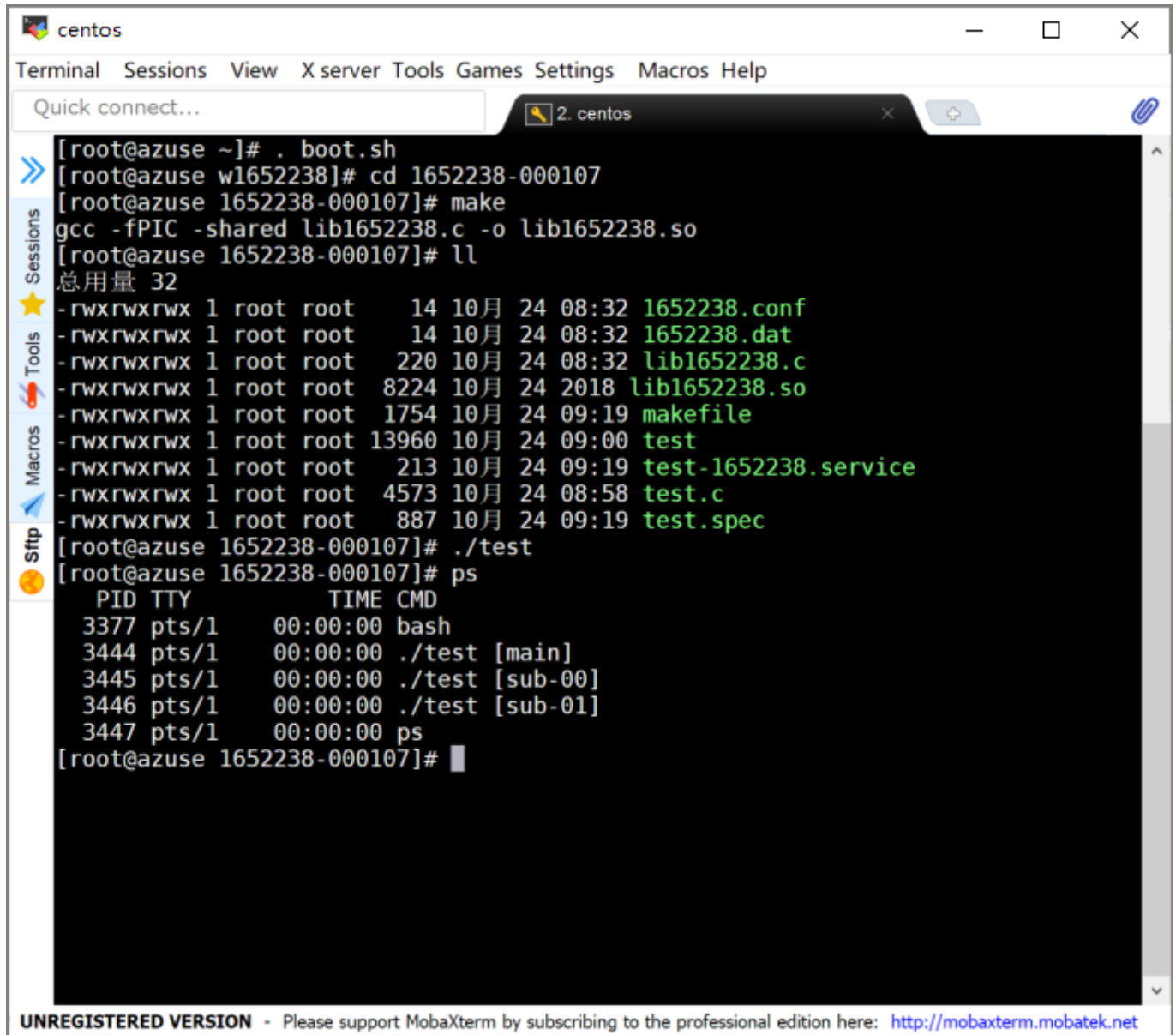
    //fork twice
    pid = fork();
    if (pid == 0)
    {
    }
    else
    {
        FILE *fp;
        fp = fopen("/var/run/test-1652238.pid", "w");
        fprintf(fp, "%d", pid);
        fclose(fp);
        exit(0);
    }

    chdir("/");
    umask(0);

    int j = open("/dev/null", O_RDWR);
    dup2(j, 0);
    dup2(j, 1);
    dup2(j, 2);

    signal(SIGCHLD, SIG_IGN);
}
```

运行，成为不脱离终端的守护进程:



The screenshot shows a MobaXterm terminal window with the following content:

```
[root@azuse ~]# . boot.sh
[root@azuse w1652238]# cd 1652238-000107
[root@azuse 1652238-000107]# make
gcc -fPIC -shared lib1652238.c -o lib1652238.so
[root@azuse 1652238-000107]# ll
总用量 32
-rwxrwxrwx 1 root root 14 10月 24 08:32 1652238.conf
-rwxrwxrwx 1 root root 14 10月 24 08:32 1652238.dat
-rwxrwxrwx 1 root root 220 10月 24 08:32 lib1652238.c
-rwxrwxrwx 1 root root 8224 10月 24 2018 lib1652238.so
-rwxrwxrwx 1 root root 1754 10月 24 09:19 makefile
-rwxrwxrwx 1 root root 13960 10月 24 09:00 test
-rwxrwxrwx 1 root root 213 10月 24 09:19 test-1652238.service
-rwxrwxrwx 1 root root 4573 10月 24 08:58 test.c
-rwxrwxrwx 1 root root 887 10月 24 09:19 test.spec
[root@azuse 1652238-000107]# ./test
[root@azuse 1652238-000107]# ps
  PID TTY          TIME CMD
  3377 pts/1        00:00:00 bash
  3444 pts/1        00:00:00 ./test [main]
  3445 pts/1        00:00:00 ./test [sub-00]
  3446 pts/1        00:00:00 ./test [sub-01]
  3447 pts/1        00:00:00 ps
[root@azuse 1652238-000107]#
```

At the bottom of the terminal window, there is a message: **UNREGISTERED VERSION** - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

2 主进程main分裂出n个子进程

使用fork分裂子进程

```

[root@azuse 1652238-000107]# ps
  PID TTY          TIME CMD
 3004 pts/0    00:00:00 bash
 3225 pts/0    00:00:00 ./test [main]
 3226 pts/0    00:00:00 ./test [sub-00]
 3227 pts/0    00:00:00 ./test [sub-01]
 3229 pts/0    00:00:00 ./test [sub-02]
 3230 pts/0    00:00:00 ./test [sub-03]
 3231 pts/0    00:00:00 ./test [sub-04]
 3232 pts/0    00:00:00 ./test [sub-05]
 3233 pts/0    00:00:00 ./test [sub-06]
 3234 pts/0    00:00:00 ./test [sub-07]
 3235 pts/0    00:00:00 ./test [sub-08]
 3236 pts/0    00:00:00 ./test [sub-09]
 3240 pts/0    00:00:00 ps
[root@azuse 1652238-000107]#

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

3 修改进程名,每秒更新自己的运行时间

- (1) 修改ps -ef现实的需要先修改argv[0]，因为argv[0]原来大小只有16字节所以要重新分配并且搬运environ。
- (2) 修改ps显示的结果使用prctl(PR_SET_NAME, comm_name);函数，最长16字节。

重新分配argv空间的函数：

```

void setproctitle_init(int argc, char **argv, char **envp)
{
    int i;

    for (i = 0; envp[i] != NULL; i++) // calc envp num
        continue;
    environ = (char **)malloc(sizeof(char *) * (i + 1)); // malloc envp
    pointer

    for (i = 0; envp[i] != NULL; i++)
    {
        environ[i] = malloc(sizeof(char) * strlen(envp[i]));
    }
}

```

```

        strcpy(environ[i], envp[i]);
    }
    environ[i] = NULL;

    g_main_Argv = argv;
    if (i > 0)
        g_main_LastArgv = envp[i - 1] + strlen(envp[i - 1]);
    else
        g_main_LastArgv = argv[argc - 1] + strlen(argv[argc - 1]);
}

```

在main函数中:

```

char argv_buf[MAXLINE] = {0}; // save argv paramters
int i;
for (i = 1; i < argc; i++)
{
    strcat(argv_buf, argv[i]);
    strcat(argv_buf, " ");
}
// 修改argv[0]所指向的内存空间的内容
setproctitle_init(argc, argv, environ);

char pid_name[MAXLINE];
char comm_name[MAXLINE];
time(&now_t);
diff_t = difftime(now_t, start_t);
snprintf(pid_name, 40, "%s [main %02d:%02d:%02d]", pid_name_origin, diff_t
/ 3600, (diff_t % 3600 - diff_t % 60) / 60, diff_t % 60);
snprintf(comm_name, 16, "%s [main]", pid_name_origin);
prctl(PR_SET_NAME, comm_name);
strcpy(g_main_Argv[0], pid_name);

```

改名效果:

```
ps -ef
```

centos

Terminal Sessions View X server Tools Games Settings Macros Help

Quick connect...

2. centos

Sessions

Tools

Macros

Sftp

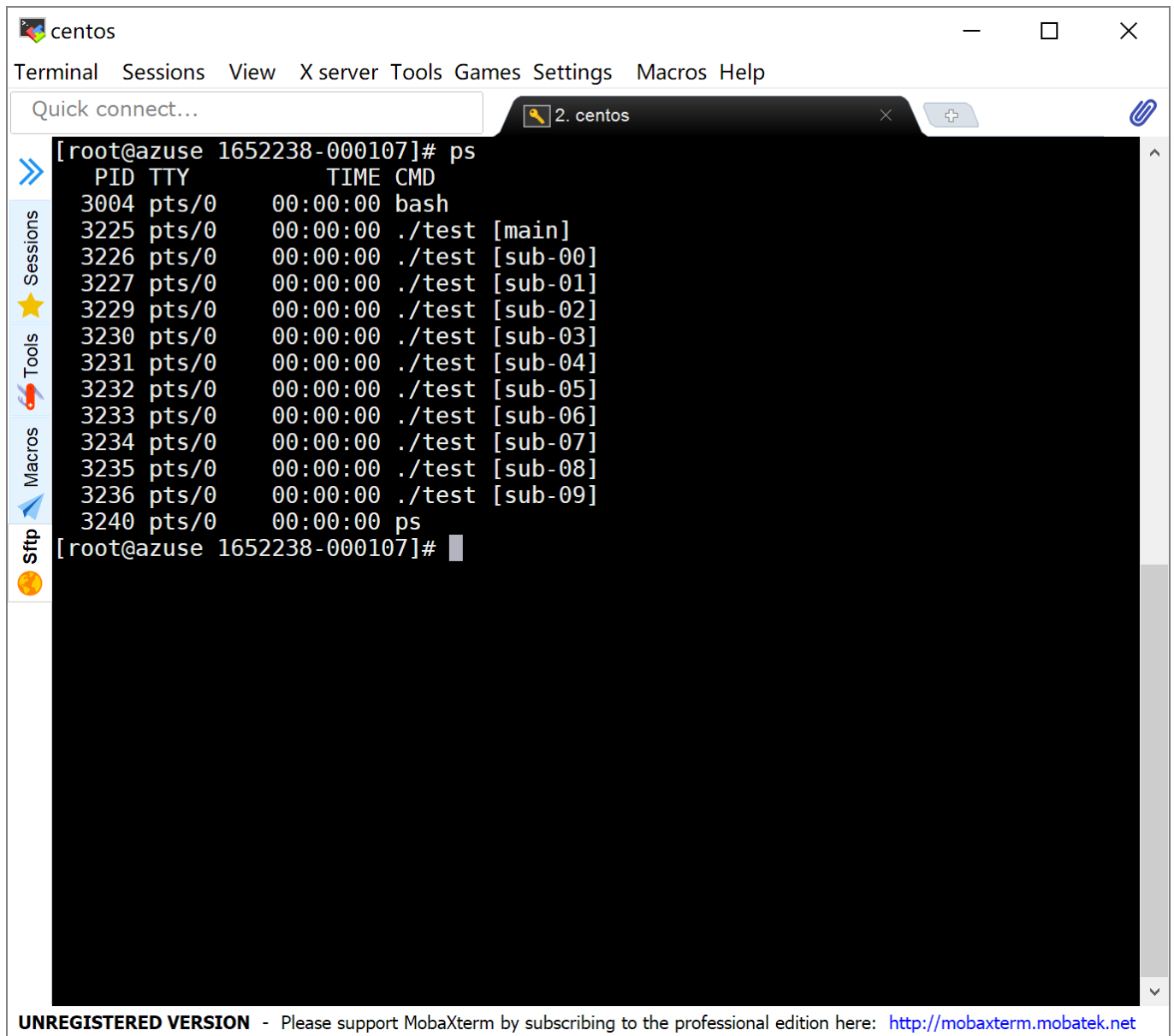
```

root      2490      1  0 16:25 ?      00:00:00 /usr/libexec/postfix/master -
postfix   2515     2490  0 16:25 ?      00:00:00 pickup -l -t unix -u
postfix   2516     2490  0 16:25 ?      00:00:00 qmgr -l -t unix -u
root      2812      1  0 16:25 ?      00:00:00 /usr/libexec/ipsec/pluto --le
root      2978      1  0 16:25 ?      00:00:00 /usr/sbin/xl2tpd -D
apache    2989     1761  0 16:25 ?      00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    2990     1761  0 16:25 ?      00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    2991     1761  0 16:25 ?      00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    2992     1761  0 16:25 ?      00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    2993     1761  0 16:25 ?      00:00:00 /usr/sbin/httpd -DFOREGROUND
root      3000     1740  0 16:26 ?      00:00:00 sshd: root@pts/0
root      3002     1740  0 16:26 ?      00:00:00 sshd: root@notty
root      3004     3000  0 16:26 pts/0    00:00:00 -bash
root      3049      1  0 16:26 ?      00:00:00 vmhgfs-fuse -o nonempty .host
root      3056     3002  0 16:26 ?      00:00:00 /usr/libexec/openssh/sftp-ser
root      3082      2  0 16:29 ?      00:00:00 [kworker/0:0]
root      3088      2  0 16:30 ?      00:00:00 [kworker/0:1]
root      3120      2  0 16:35 ?      00:00:00 [kworker/0:2]
root      3210      2  0 16:38 ?      00:00:00 [kworker/0:3]
root      3225      1  0 16:41 pts/0    00:00:00 ./test [main 00:00:22]
root      3226     3225  0 16:41 pts/0    00:00:00 ./test [sub-00 00:00:22]
root      3227     3225  0 16:41 pts/0    00:00:00 ./test [sub-01 00:00:21]
root      3229     3225  0 16:41 pts/0    00:00:00 ./test [sub-02 00:00:20]
root      3230     3225  0 16:41 pts/0    00:00:00 ./test [sub-03 00:00:19]
root      3231     3225  0 16:41 pts/0    00:00:00 ./test [sub-04 00:00:18]
root      3232     3225  0 16:41 pts/0    00:00:00 ./test [sub-05 00:00:17]
root      3233     3225  0 16:41 pts/0    00:00:00 ./test [sub-06 00:00:16]
root      3234     3225  0 16:41 pts/0    00:00:00 ./test [sub-07 00:00:15]
root      3235     3225  0 16:41 pts/0    00:00:00 ./test [sub-08 00:00:14]
root      3236     3225  0 16:41 pts/0    00:00:00 ./test [sub-09 00:00:13]
root      3238     3004  0 16:41 pts/0    00:00:00 ps -ef
[root@azuse 1652238-000107]#

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

ps



```
[root@azuse 1652238-000107]# ps
  PID TTY          TIME CMD
 3004 pts/0        00:00:00 bash
 3225 pts/0        00:00:00 ./test [main]
 3226 pts/0        00:00:00 ./test [sub-00]
 3227 pts/0        00:00:00 ./test [sub-01]
 3229 pts/0        00:00:00 ./test [sub-02]
 3230 pts/0        00:00:00 ./test [sub-03]
 3231 pts/0        00:00:00 ./test [sub-04]
 3232 pts/0        00:00:00 ./test [sub-05]
 3233 pts/0        00:00:00 ./test [sub-06]
 3234 pts/0        00:00:00 ./test [sub-07]
 3235 pts/0        00:00:00 ./test [sub-08]
 3236 pts/0        00:00:00 ./test [sub-09]
 3240 pts/0        00:00:00 ps
[root@azuse 1652238-000107]#
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

4 kill子进程后父进程自动补全

这个实现方法有很多，我是用记录下所有子进程的pid，让父进程每次sleep醒来后检查一遍子进程是不是都还在，如果有子进程不在了就重新补齐。

在main的while循环中，用kill命令循环对每个子进程发0信号，若进程pid不存在就判定为子进程被kill，fork出新的子进程后，让子进程进入死循环，main继续循环检查。

```
for (i = 0; i < 10; i++)
{
    if (kill(myson[i], 0) == -1)
    {
        pid = fork();
        if (pid == 0)
        {
            prctl(PR_SET_PDEATHSIG, SIGHUP);
            time(&start_t);
            break;
        }
    }
}
```

```

    }
    else
    {
        myson[i] = pid;
    }
}
}

```

使用效果：kill之后自动恢复子函数

The screenshot shows a MobaXterm terminal window with the following content:

```

[centos]
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
2. centos
[root@azuse 1652238-000107]# ps
  PID TTY          TIME CMD
 3004 pts/0        00:00:00 bash
 3191 pts/0        00:00:00 ./test [main]
 3192 pts/0        00:00:00 ./test [sub-00]
 3193 pts/0        00:00:00 ./test [sub-01]
 3195 pts/0        00:00:00 ./test [sub-02]
 3198 pts/0        00:00:00 ./test [sub-03]
 3200 pts/0        00:00:00 ./test [sub-04]
 3202 pts/0        00:00:00 ./test [sub-05]
 3204 pts/0        00:00:00 ./test [sub-06]
 3205 pts/0        00:00:00 ./test [sub-07]
 3206 pts/0        00:00:00 ./test [sub-08]
 3207 pts/0        00:00:00 ./test [sub-09]
 3209 pts/0        00:00:00 ps
[root@azuse 1652238-000107]# kill 3192 3193
[root@azuse 1652238-000107]# ps
  PID TTY          TIME CMD
 3004 pts/0        00:00:00 bash
 3191 pts/0        00:00:00 ./test [main]
 3195 pts/0        00:00:00 ./test [sub-02]
 3198 pts/0        00:00:00 ./test [sub-03]
 3200 pts/0        00:00:00 ./test [sub-04]
 3202 pts/0        00:00:00 ./test [sub-05]
 3204 pts/0        00:00:00 ./test [sub-06]
 3205 pts/0        00:00:00 ./test [sub-07]
 3206 pts/0        00:00:00 ./test [sub-08]
 3207 pts/0        00:00:00 ./test [sub-09]
 3211 pts/0        00:00:00 ./test [sub-00]
 3212 pts/0        00:00:00 ./test [sub-01]
 3213 pts/0        00:00:00 ps
[root@azuse 1652238-000107]#
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

```

5 从/etc/1652238.conf中读取子进程数量=

读取config文件的函数放在lib1652238.c中

使用gcc -fPIC -shared lib1652238.c -o lib1652238.so编译成共享库

使用gcc test.c -L. -l1652238 -o test使用共享库编译test

再把编译好的lib1652238.so放到/usr/lib64中

```
//lib1652238.so
#include <stdio.h>
#include <stdlib.h>
int readconf()
{
    int i;
    FILE *fp;
    fp = fopen("/etc/1652238.conf", "r");
    fscanf(fp, "子进程数量=%d", &i);
    fclose(fp);

a'w'd
    if (i > 20 || i < 5)
        i = 5;
    return i;
}
```

6 makefile文件和rpm文件编写

makefile install需要放置可执行文件，动态链接库，配置文件和数据文件还有service文件
if分支是为了避免在编译rpm文件的过程中误启动systemctl

```
install : all
    mkdir -p $(DIRROOT)/usr $(DIRROOT)/usr/sbin $(DIRROOT)/usr/lib64
    $(DIRROOT)/etc $(DIRROOT)/usr/lib $(DIRROOT)/usr/lib/systemd
    $(DIRROOT)/usr/lib/systemd/system
    cp test $(DIRROOT)/usr/sbin/$(EXECNAME)
    cp lib1652238.so $(DIRROOT)/usr/lib64
    mkdir -p $(DIRROOT)/usr/1652238
    cp 1652238.dat $(DIRROOT)/usr/1652238
    cp 1652238.conf $(DIRROOT)/etc
    cp test-1652238.service $(DIRROOT)/usr/lib/systemd/system/

ifeq ($(DIRROOT), )
    systemctl daemon-reload
    systemctl enable test-1652238
    systemctl start test-1652238
endif
```

makeuninstall需要移除上述文件，注意要先停止服务确保程序不在运行后再移除

```
uninstall :
    systemctl stop test-1652238
    rm -f $(DIRROOT)/usr/sbin/$(EXECNAME) $(DIRROOT)/usr/lib64/lib1652238
    $(DIRROOT)/etc/1652238.conf $(DIRROOT)/usr/1652238/1652238.dat
    rm -df $(DIRROOT)/usr/1652238
    rm -f *.o test lib1652238.so
    rm -f $(DIRROOT)/usr/lib/systemd/system/test-1652238.service
```


rpm包编写 先编写.spec文件

```
Name:          test-1652238
Version:       1.0.0
Release:       1%{?dist}
Summary:       test-1652238
License:       GPL
Packager:      abel
Source0:       %_sourcedir/test.tar.bz2
%description
%prep
%setup -q
%build
make
%install
make install DIRROOT=%{buildroot}
%pre
echo "准备安装 test-1652238"
%post
systemctl daemon-reload
systemctl enable test-1652238
systemctl start test-1652238
echo "完成安装 test-1652238s"
%preun
echo "准备卸载 test-1652238"
systemctl stop test-1652238
%postun
echo "完成卸载 test-1652238"
%files
%{_sbindir}/test-1652238
%{_libdir}/lib1652238.so
%{_prefix}/1652238/1652238.dat
%{_sysconfdir}/1652238.conf
%{_unitdir}/test-1652238.service
%changelog
```

然后再makefile中创建好源码包，复制到SOURCE中，再让rpm build时make install到BUILDDROOT下，完成编译后清楚rpm临时目录

用`--define "_topdir ${CURDIR}"`来让rpm在当前目录编译

```
rpm      :
          mkdir -p SOURCES
          mkdir -p BUILD
          mkdir -p BUILDDROOT
          mkdir -p RPMS
          mkdir -p SRPMS
          mkdir -p test-1652238-1.0.0
          cp makefile test.c lib1652238.c 1652238.dat 1652238.conf test-
1652238.service test-1652238-1.0.0/
          # cp makefile test.c lib1652238.c 1652238.dat 1652238.conf test-
```

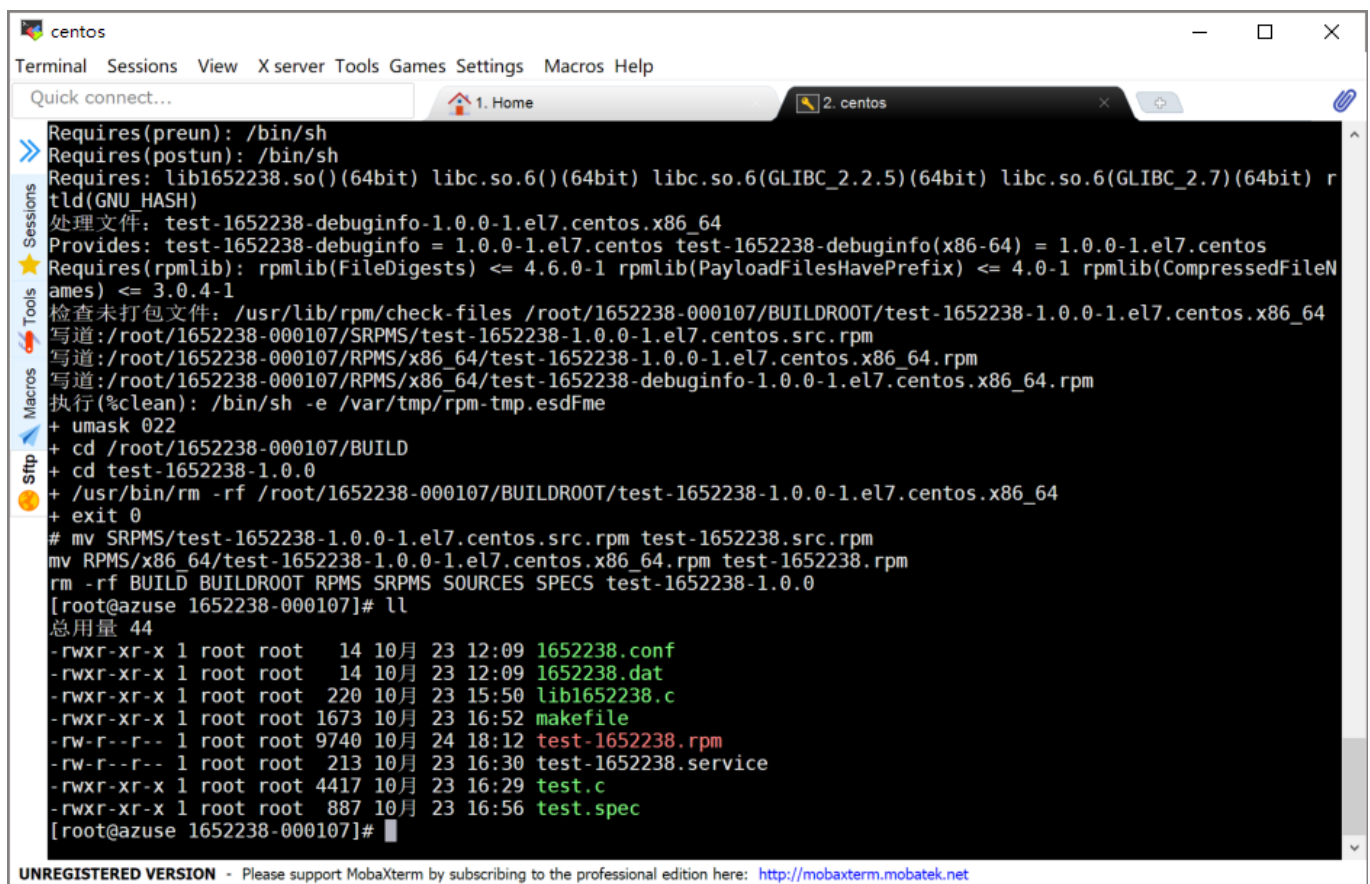
```
1652238.service SOURCES/
tar -cjf test.tar.bz2 test-1652238-1.0.0
mv test.tar.bz2 SOURCES

rpmbuild -ba test.spec --define "_topdir ${CURDIR}"

# mv SRPMS/test-1652238-1.0.0-1.el7.centos.src.rpm test-1652238.src.rpm
mv RPMS/x86_64/test-1652238-1.0.0-1.el7.centos.x86_64.rpm test-1652238.rpm

rm -rf BUILD BUILDROOT RPMS SRPMS SOURCES SPECS test-1652238-1.0.0
```

正常生成rpm



```
centos
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect... 1. Home 2. centos
Requires(preun): /bin/sh
Requires(postun): /bin/sh
Requires: lib1652238.so()(64bit) libc.so.6()(64bit) libc.so.6(GLIBC_2.2.5)(64bit) libc.so.6(GLIBC_2.7)(64bit) r
tld(GNU_HASH)
处理文件: test-1652238-debuginfo-1.0.0-1.el7.centos.x86_64
Provides: test-1652238-debuginfo = 1.0.0-1.el7.centos test-1652238-debuginfo(x86-64) = 1.0.0-1.el7.centos
Requires(rpmlib): rpmlib(FileDigests) <= 4.6.0-1 rpmlib(PayloadFilesHavePrefix) <= 4.0-1 rpmlib(CompressedFileN
ames) <= 3.0.4-1
检查未打包文件: /usr/lib/rpm/check-files /root/1652238-000107/BUILDROOT/test-1652238-1.0.0-1.el7.centos.x86_64
写道:/root/1652238-000107/SRPMS/test-1652238-1.0.0-1.el7.centos.src.rpm
写道:/root/1652238-000107/RPMS/x86_64/test-1652238-1.0.0-1.el7.centos.x86_64.rpm
写道:/root/1652238-000107/RPMS/x86_64/test-1652238-debuginfo-1.0.0-1.el7.centos.x86_64.rpm
执行(%clean): /bin/sh -e /var/tmp/rpm-tmp.esdFme
+ umask 022
+ cd /root/1652238-000107/BUILD
+ cd test-1652238-1.0.0
+ /usr/bin/rm -rf /root/1652238-000107/BUILDROOT/test-1652238-1.0.0-1.el7.centos.x86_64
+ exit 0
# mv SRPMS/test-1652238-1.0.0-1.el7.centos.src.rpm test-1652238.src.rpm
mv RPMS/x86_64/test-1652238-1.0.0-1.el7.centos.x86_64.rpm test-1652238.rpm
rm -rf BUILD BUILDROOT RPMS SRPMS SOURCES SPECS test-1652238-1.0.0
[root@azuse 1652238-000107]# ll
总用量 44
-rwxr-xr-x 1 root root 14 10月 23 12:09 1652238.conf
-rwxr-xr-x 1 root root 14 10月 23 12:09 1652238.dat
-rwxr-xr-x 1 root root 220 10月 23 15:50 lib1652238.c
-rwxr-xr-x 1 root root 1673 10月 23 16:52 makefile
-rw-r--r-- 1 root root 9740 10月 24 18:12 test-1652238.rpm
-rw-r--r-- 1 root root 213 10月 23 16:30 test-1652238.service
-rwxr-xr-x 1 root root 4417 10月 23 16:29 test.c
-rwxr-xr-x 1 root root 887 10月 23 16:56 test.spec
[root@azuse 1652238-000107]#
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

rpm正常安装与卸载(注意卸载时必须用test-1652238)

```

centos
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
1. Home 2. centos
[root@azuse 1652238-000107]# rpm -ihv test-1652238.rpm
准备中... ##### [100%]
准备安装 test-1652238
正在升级/安装...
1:test-1652238-1.0.0-1.el7.centos ##### [100%]
完成安装 test-1652238s
[root@azuse 1652238-000107]# rpm -ehv test-1652238
准备中... ##### [100%]
准备卸载 test-1652238
正在清理/删除...
1:test-1652238-1.0.0-1.el7.centos ##### [100%]
完成卸载 test-1652238
[root@azuse 1652238-000107]#

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

make clean部分

清除可执行文件和动态链接库

清除rpm包

clean:

```
rm -rf BUILD BUILDROOT RPMS SRPMS SOURCES SPECS test-1652238-1.0.0
rm -f *.o test lib1652238.so *.rpm
```

7 service文件编写

service文件编写比较简单，难点在于模式设置成forking之后systemctl会默认把第二次fork的中间进程作为父进程，无法正常启动，所以程序启动时自己去写个pid文件

```

[Unit]
Description=test-1652238.service

[Service]
Type=forking
ExecStart=/usr/sbin/test-1652238
PIDFile=/var/run/test-1652238.pid
StandardOutput=syslog
StandardError=inherit

[Install]
WantedBy=multi-user.target

```

在`daemon_by_hand()`函数中写pid的方法:

```
//fork twice
pid = fork();
if (pid == 0)
{
}
else
{
    FILE *fp;
    fp = fopen("/var/run/test-1652238.pid", "w");
    fprintf(fp, "%d", pid);
    fclose(fp);
    exit(0);
}
```

systemctl正常使用

```
centos
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect... 1. Home 2. centos
[root@azuse 1652238-000107]# systemctl start test-1652238
[root@azuse 1652238-000107]# systemctl stuts test-1652238
Unknown operation 'stuts'.
[root@azuse 1652238-000107]# systemctl status test-1652238
* test-1652238.service
   Loaded: loaded (/usr/lib/systemd/system/test-1652238.service; enabled; vendor preset: disabled)
   Active: active (running) since 三 2018-10-24 18:31:03 CST; 29s ago
   Process: 4768 ExecStart=/usr/sbin/test-1652238 (code=exited, status=0/SUCCESS)
   Main PID: 4770 (test [main])
   CGroup: /system.slice/test-1652238.service
           |-4770 test [main 00:00:29]
           |-4771 test [sub-00 00:00:29]
           |-4772 test [sub-01 00:00:28]
           |-4774 test [sub-02 00:00:27]
           |-4775 test [sub-03 00:00:26]
           |-4776 test [sub-04 00:00:25]
           |-4777 test [sub-05 00:00:24]
           |-4778 test [sub-06 00:00:23]
           |-4779 test [sub-07 00:00:22]
           |-4793 test [sub-08 00:00:21]
           |-4798 test [sub-09 00:00:20]
  10月 24 18:31:03 azuse.centos systemd[1]: Starting test-1652238.service...
  10月 24 18:31:03 azuse.centos systemd[1]: PID file /var/run/test-1652238.pid not readable (yet?) after start.
  10月 24 18:31:03 azuse.centos systemd[1]: Started test-1652238.service.
[root@azuse 1652238-000107]# systemctl stop test-1652238
[root@azuse 1652238-000107]# systemctl status test-1652238
* test-1652238.service
   Loaded: loaded (/usr/lib/systemd/system/test-1652238.service; enabled; vendor preset: disabled)
   Active: inactive (dead) since 三 2018-10-24 18:31:48 CST; 2s ago
   Process: 4768 ExecStart=/usr/sbin/test-1652238 (code=exited, status=0/SUCCESS)
   Main PID: 4770 (code=killed, signal=TERM)
  10月 24 18:31:03 azuse.centos systemd[1]: Starting test-1652238.service...
  10月 24 18:31:03 azuse.centos systemd[1]: PID file /var/run/test-1652238.pid not readable (yet?) after start.
  10月 24 18:31:03 azuse.centos systemd[1]: Started test-1652238.service.
  10月 24 18:31:48 azuse.centos systemd[1]: Stopping test-1652238.service...
  10月 24 18:31:48 azuse.centos systemd[1]: Stopped test-1652238.service.
[root@azuse 1652238-000107]#
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>