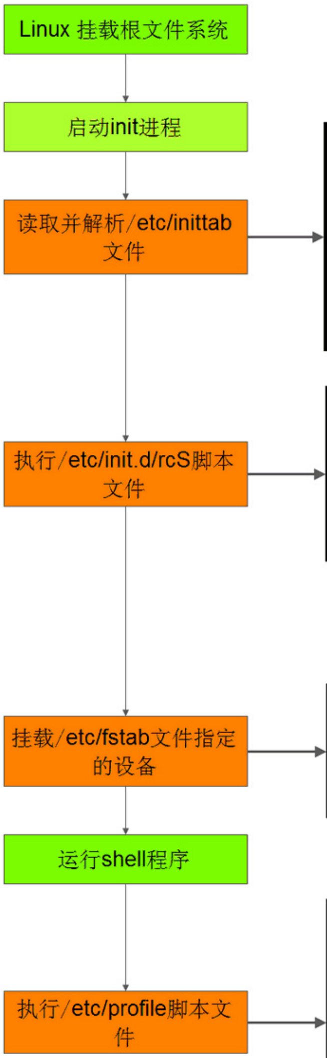
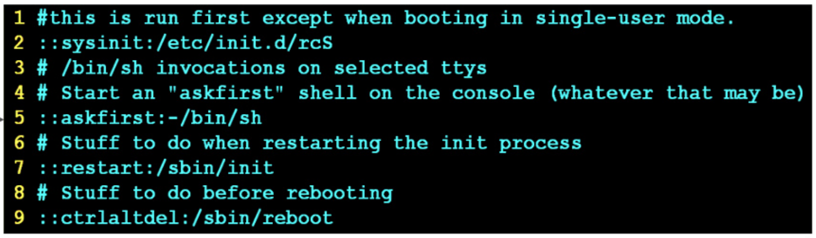
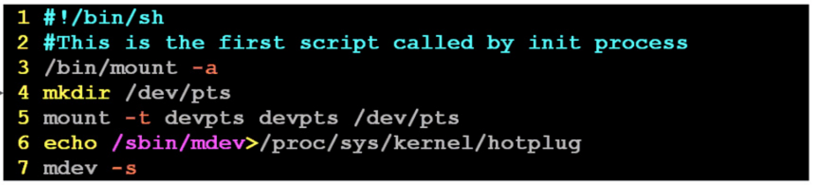
Linux 系统挂载完根文件系统后，就会执行init程序，创建init进程。init进程在运行的过程中会读取/etc/inittab文件,然后进行解析，执行需要执行的应用程序和脚本文件。大概流程如下:



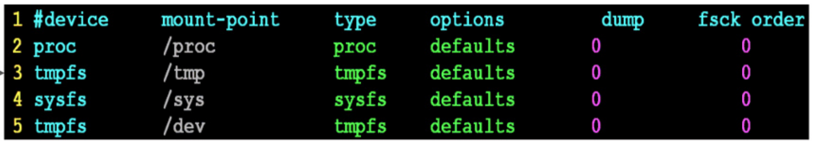
Linux 挂载根文件系统
启动init进程
读取并解析/etc/inittab文件
执行/etc/init.d/rcS脚本文件
挂载/etc/fstab文件指定的设备
运行shell程序
执行/etc/profile脚本文件



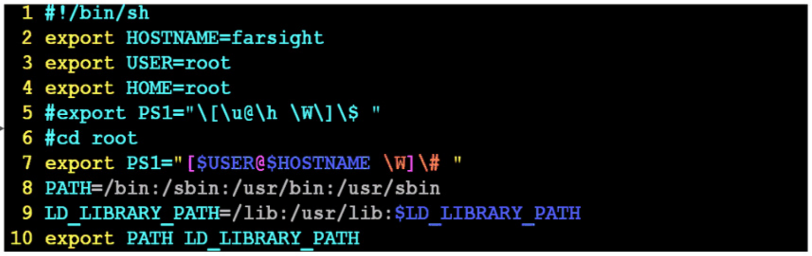
```
#this is run first except when booting in single-user mode.
::sysinit:/etc/init.d/rcS
# /bin/sh invocations on selected ttys
# Start an "askfirst" shell on the console (whatever that may be)
::askfirst:-/bin/sh
# Stuff to do when restarting the init process
::restart:/sbin/init
# Stuff to do before rebooting
::ctrlaltdel:/sbin/reboot
```



```
#!/bin/sh
#This is the first script called by init process
/bin/mount -a
mkdir /dev/pts
mount -t devpts devpts /dev/pts
echo /sbin/mdev>/proc/sys/kernel/hotplug
mdev -s
```



```
#device     mount-point     type      options        dump     fsck order
proc        /proc           proc      defaults       0          0
tmpfs       /tmp            tmpfs     defaults       0          0
sysfs       /sys            sysfs     defaults       0          0
tmpfs       /dev            tmpfs     defaults       0          0
```

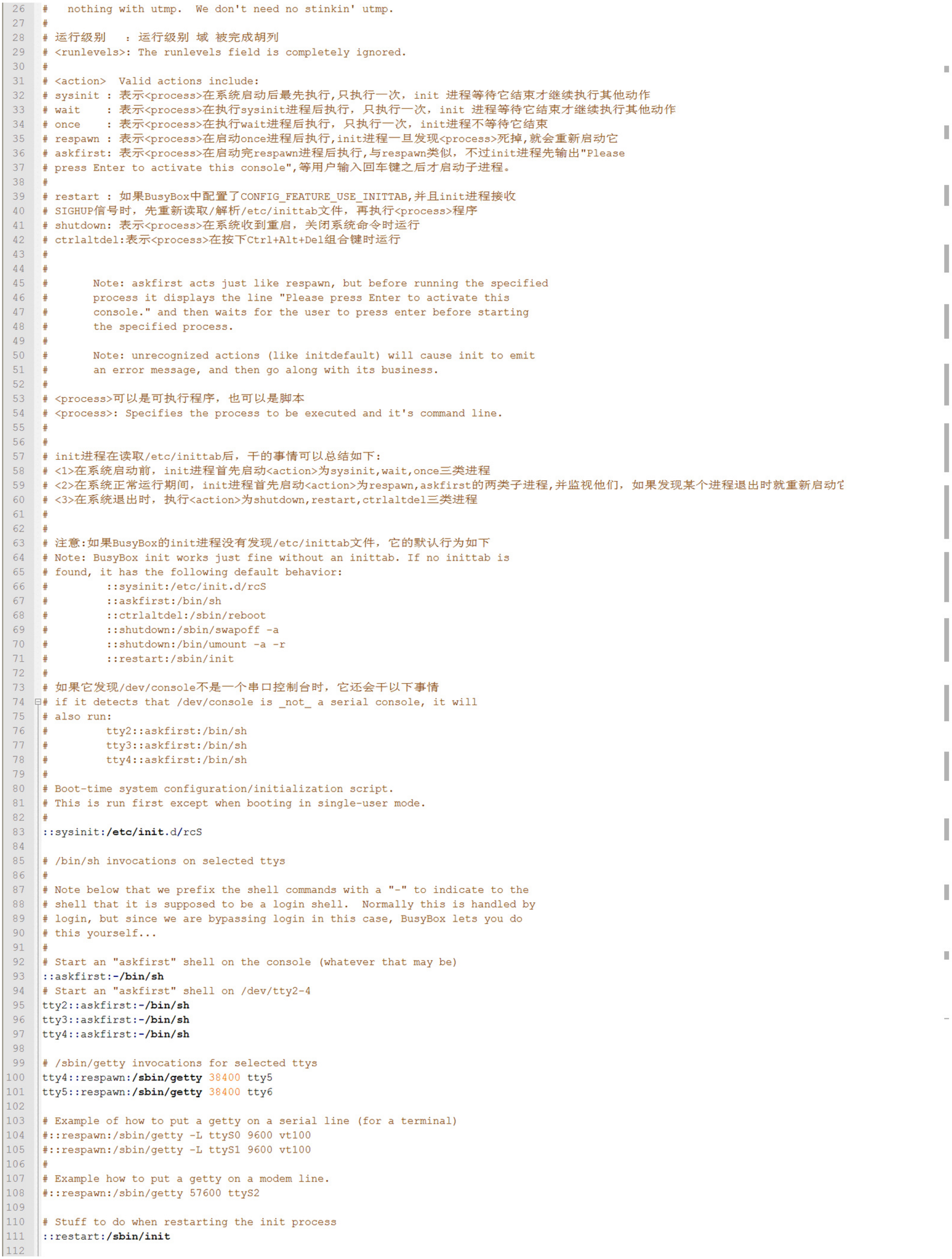


```
#!/bin/sh
export HOSTNAME=farsight
export USER=root
export HOME=root
#export PS1="\[\u@\h \W\]\$ "
#cd root
export PS1="[$USER@$HOSTNAME \W]\# "
PATH=/bin:/sbin:/usr/bin:/usr/sbin
LD_LIBRARY_PATH=/lib:/usr/lib:$LD_LIBRARY_PATH
export PATH LD_LIBRARY_PATH
```

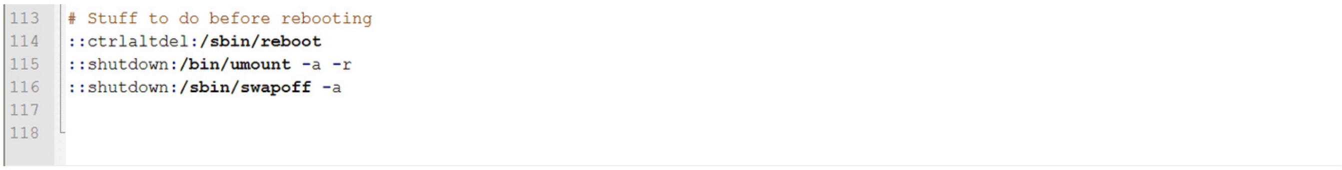
接下来我们分析一下，每个文件所包含内容的具体含义。

一、/etc/inittab 文件内容分析

```
# /etc/inittab init(8) configuration for BusyBox
#
# Copyright (C) 1999-2004 by Erik Andersen <andersen@codepoet.org>
#
# 注意，BusyBox的init的程序是不支持运行级别的。   这里的运行级别域
# Note, BusyBox init doesn't support runlevels.  The runlevels field is
# 会被BusyBox的init程序完全忽略        如果你想使用运行级别，可以使用sysvinit
# completely ignored by BusyBox init. If you want runlevels, use sysvinit.
#
#
# /etc/inittab文件中每个条目的格式
# Format for each entry:<id>:<runlevels>:<action>:<process>
#
# 警告:<id>的这个域，对于BusyBox的init进程而言，它的含义和传统的含义不一样
# 即不是ID的意思,这里的ID表示它后面跟的程序运行的时候使用的控制台，如果这个
# 这个字段忽略，则使用和init进程一样的控制台
# <id>: WARNING: This field has a non-traditional meaning for BusyBox init!
#
#   The id field is used by BusyBox init to specify the controlling tty for
#   the specified process to run on.The contents of this field are
#   appended to "/dev/" and used as-is.  There is no need for this field to
#   be unique, although if it isn't you may have strange results.  If this
#   field is left blank, it is completely ignored.  Also note that if
#   BusyBox detects that a serial console is in use, then all entries
#   containing non-empty id fields will be ignored.  BusyBox init does
```

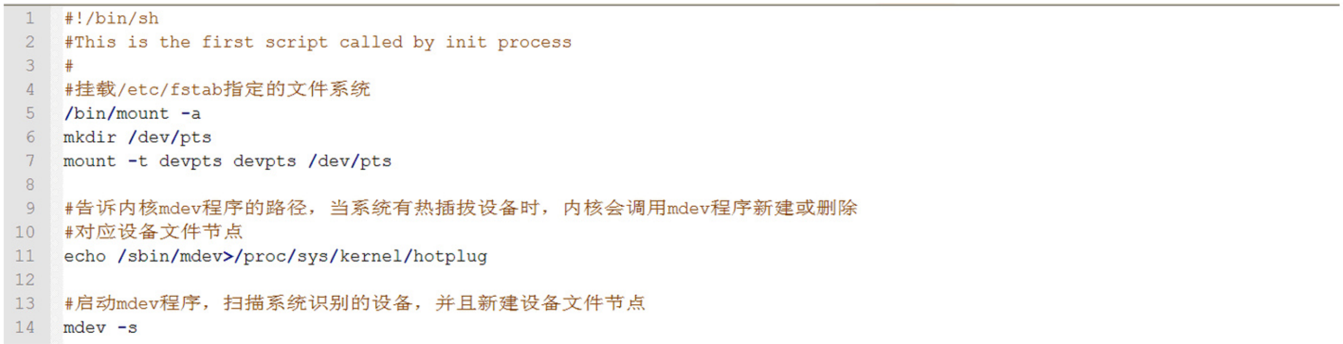


```
#   nothing with utmp.  We don't need no stinkin' utmp.
#
# 运行级别   : 运行级别 域 被完成胡列
# <runlevels>: The runlevels field is completely ignored.
#
# <action>  Valid actions include:
# sysinit : 表示<process>在系统启动后最先执行,只执行一次，init 进程等待它结束才继续执行其他动作
# wait    : 表示<process>在执行sysinit进程后执行，只执行一次，init 进程等待它结束才继续执行其他动作
# once    : 表示<process>在执行wait进程后执行，只执行一次，init进程不等待它结束
# respawn : 表示<process>在启动once进程后执行,init进程一旦发现<process>死掉,就会重新启动它
# askfirst: 表示<process>在启动完respawn进程后执行,与respawn类似，不过init进程先输出"Please
# press Enter to activate this console",等用户输入回车键之后才启动子进程。
#
# restart : 如果BusyBox中配置了CONFIG_FEATURE_USE_INITTAB,并且init进程接收
# SIGHUP信号时，先重新读取/解析/etc/inittab文件，再执行<process>程序
# shutdown: 表示<process>在系统收到重启，关闭系统命令时运行
# ctrlaltdel:表示<process>在按下Ctrl+Alt+Del组合键时运行
#
#
#       Note: askfirst acts just like respawn, but before running the specified
#       process it displays the line "Please press Enter to activate this
#       console." and then waits for the user to press enter before starting
#       the specified process.
#
#       Note: unrecognized actions (like initdefault) will cause init to emit
#       an error message, and then go along with its business.
#
# <process>可以是可执行程序，也可以是脚本
# <process>: Specifies the process to be executed and it's command line.
#
#
# init进程在读取/etc/inittab后，干的事情可以总结如下:
# <1>在系统启动前，init进程首先启动<action>为sysinit,wait,once三类进程
# <2>在系统正常运行期间，init进程首先启动<action>为respawn,askfirst的两类子进程,并监视他们，如果发现某个进程退出时就重新启动它
# <3>在系统退出时，执行<action>为shutdown,restart,ctrlaltdel三类进程
#
#
# 注意:如果BusyBox的init进程没有发现/etc/inittab文件，它的默认行为如下
# Note: BusyBox init works just fine without an inittab. If no inittab is
# found, it has the following default behavior:
#         ::sysinit:/etc/init.d/rcS
#         ::askfirst:/bin/sh
#         ::ctrlaltdel:/sbin/reboot
#         ::shutdown:/sbin/swapoff -a
#         ::shutdown:/bin/umount -a -r
#         ::restart:/sbin/init
#
# 如果它发现/dev/console不是一个串口控制台时，它还会干以下事情
# if it detects that /dev/console is _not_ a serial console, it will
# also run:
#         tty2::askfirst:/bin/sh
#         tty3::askfirst:/bin/sh
#         tty4::askfirst:/bin/sh
#
# Boot-time system configuration/initialization script.
# This is run first except when booting in single-user mode.
#
::sysinit:/etc/init.d/rcS

# /bin/sh invocations on selected ttys
#
# Note below that we prefix the shell commands with a "-" to indicate to the
# shell that it is supposed to be a login shell.  Normally this is handled by
# login, but since we are bypassing login in this case, BusyBox lets you do
# this yourself...
#
# Start an "askfirst" shell on the console (whatever that may be)
::askfirst:-/bin/sh
# Start an "askfirst" shell on /dev/tty2-4
tty2::askfirst:-/bin/sh
tty3::askfirst:-/bin/sh
tty4::askfirst:-/bin/sh

# /sbin/getty invocations for selected ttys
tty4::respawn:/sbin/getty 38400 tty5
tty5::respawn:/sbin/getty 38400 tty6

# Example of how to put a getty on a serial line (for a terminal)
#::respawn:/sbin/getty -L ttyS0 9600 vt100
#::respawn:/sbin/getty -L ttyS1 9600 vt100
#
# Example how to put a getty on a modem line.
#::respawn:/sbin/getty 57600 ttyS2

# Stuff to do when restarting the init process
::restart:/sbin/init

```



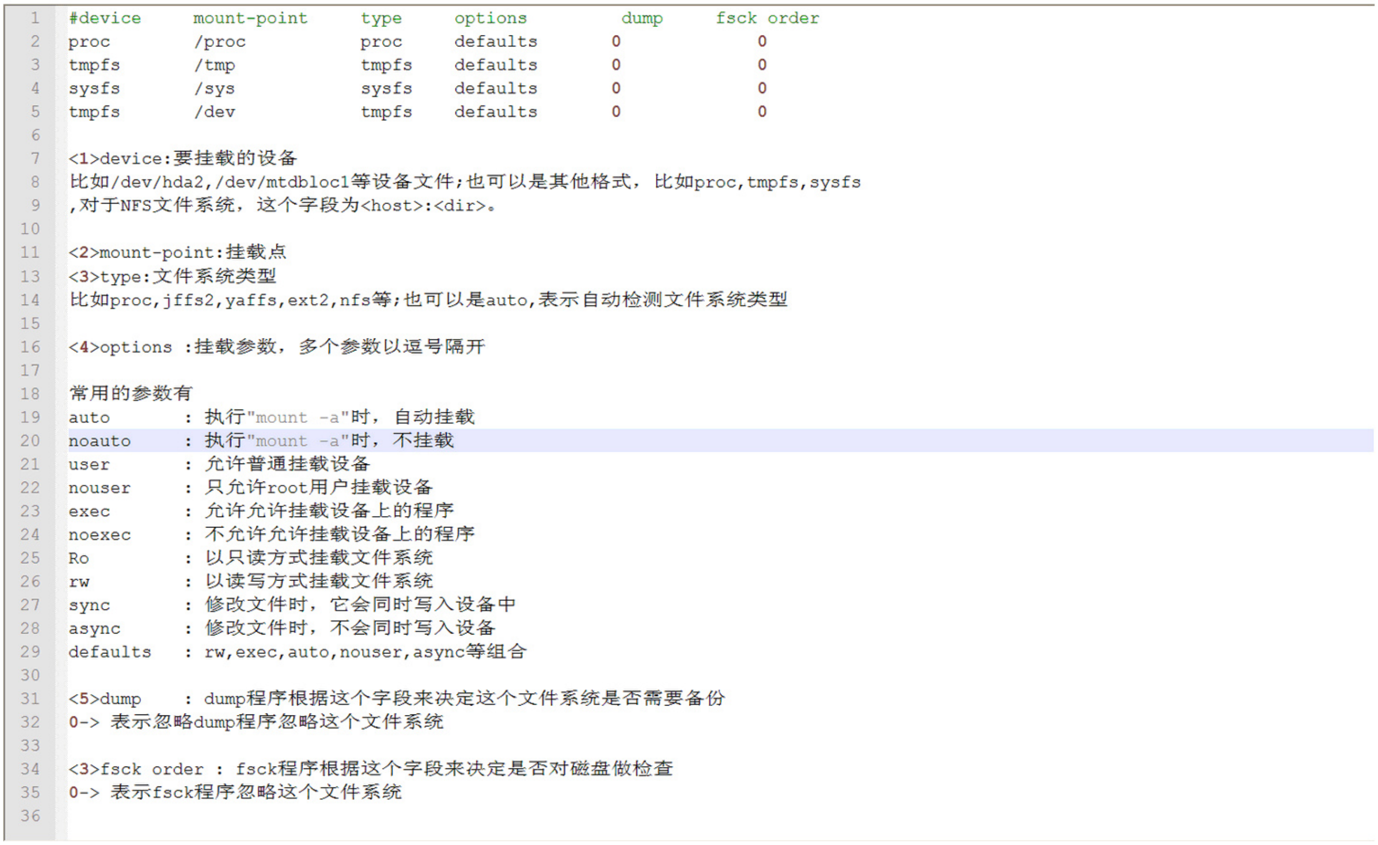
```
# Stuff to do before rebooting
::ctrlaltdel:/sbin/reboot
::shutdown:/bin/umount -a -r
::shutdown:/sbin/swapoff -a

```

二、/etc/init.d/rcS文件内容分析



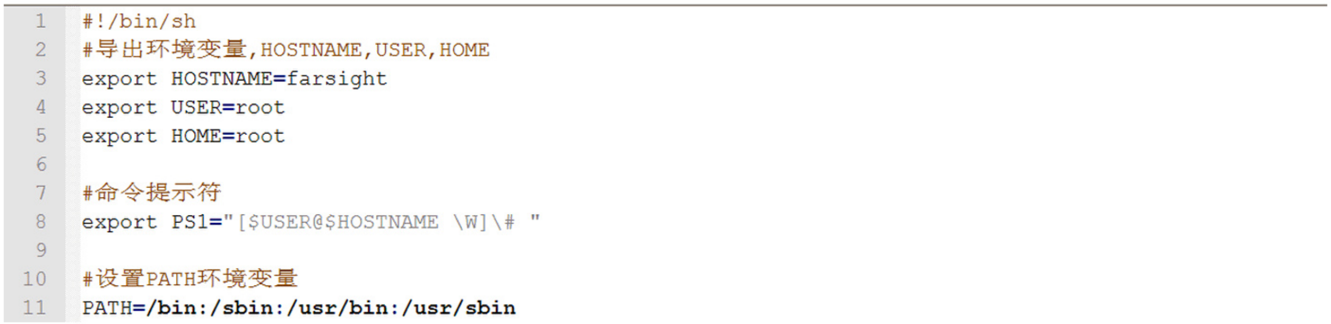
```
#!/bin/sh
#This is the first script called by init process
#
#挂载/etc/fstab指定的文件系统
/bin/mount -a
mkdir /dev/pts
mount -t devpts devpts /dev/pts

#告诉内核mdev程序的路径，当系统有热插拔设备时，内核会调用mdev程序新建或删除
#对应设备文件节点
echo /sbin/mdev>/proc/sys/kernel/hotplug

#启动mdev程序，扫描系统识别的设备，并且新建设备文件节点
mdev -s
```

三、/etc/fstab文件内容分析

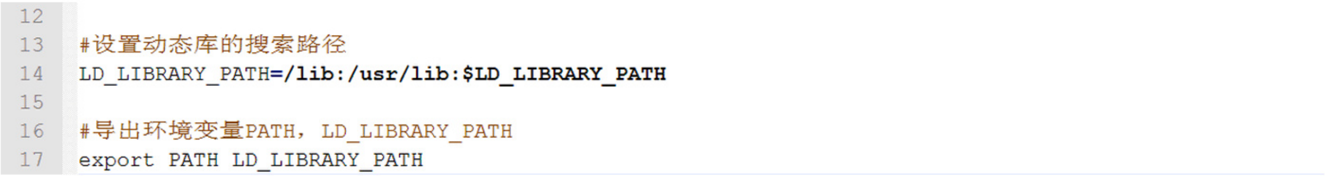


```
#device      mount-point      type     options         dump     fsck order
proc         /proc            proc     defaults       0              0
tmpfs        /tmp             tmpfs    defaults       0              0
sysfs        /sys             sysfs    defaults       0              0
tmpfs        /dev             tmpfs    defaults       0              0

<1>device:要挂载的设备
比如/dev/hda2,/dev/mtdbloc1等设备文件;也可以是其他格式，比如proc,tmpfs,sysfs
,对于NFS文件系统，这个字段为<host>:<dir>。

<2>mount-point:挂载点
<3>type:文件系统类型
比如proc,jffs2,yaffs,ext2,nfs等;也可以是auto,表示自动检测文件系统类型

<4>options :挂载参数，多个参数以逗号隔开

常用的参数有
auto        : 执行"mount -a"时，自动挂载
noauto      : 执行"mount -a"时，不挂载
user        : 允许普通挂载设备
nouser      : 只允许root用户挂载设备
exec        : 允许允许挂载设备上的程序
noexec      : 不允许允许挂载设备上的程序
Ro          : 以只读方式挂载文件系统
rw          : 以读写方式挂载文件系统
sync        : 修改文件时，它会同时写入设备中
async       : 修改文件时，不会同时写入设备
defaults    : rw,exec,auto,nouser,async等组合

<5>dump     : dump程序根据这个字段来决定这个文件系统是否需要备份
0-> 表示忽略dump程序忽略这个文件系统

<3>fsck order : fsck程序根据这个字段来决定是否对磁盘做检查
0-> 表示fsck程序忽略这个文件系统

```

四、/etc/profile文件内容分析



```
#!/bin/sh
#导出环境变量,HOSTNAME,USER,HOME
export HOSTNAME=farsight
export USER=root
export HOME=root

#命令提示符
export PS1="[$USER@$HOSTNAME \W]\# "

#设置PATH环境变量
PATH=/bin:/sbin:/usr/bin:/usr/sbin
```



```

#设置动态库的搜索路径
LD_LIBRARY_PATH=/lib:/usr/lib:$LD_LIBRARY_PATH

#导出环境变量PATH，LD_LIBRARY_PATH
export PATH LD_LIBRARY_PATH
```