

```

1  #!/bin/sh
2  #
3  # rc          This file is responsible for starting/stopping
4  #              services when the runlevel changes.
5  #
6  #              Optimization feature:
7  #              A startup script is _not_ run when the service was
8  #              running in the previous runlevel and it wasn't stopped
9  #              in the runlevel transition (most Debian services don't
10 #              have K?? links in rc{1,2,3,4,5} )
11 #
12 # Author:      Miquel van Smoorenburg <miquels@cistron.nl>
13 #              Bruce Perens <Bruce@Pixar.com>
14 #
15 # Version:     @(#)rc 2.78 07-Nov-1999 miquels@cistron.nl
16 #
17
18 . /etc/default/rcS
19 export VERBOSE
20
21 startup_progress() {
22     step=$(( $step + $step_change ))
23     if [ "$num_steps" != "0" ]; then
24         progress=$(( ($step * $progress_size / $num_steps) + $first_step ))
25     else
26         progress=$progress_size
27     fi
28     #echo "PROGRESS is $progress $runlevel $first_step + ($step of $num_steps)
29     #step_change $progress_size"
30     if type psplash-write >/dev/null 2>&1; then
31         TMPDIR=/mnt/.psplash psplash-write "PROGRESS $progress" || true
32     fi
33     #if [ -e /mnt/.psplash/psplash_fifo ]; then
34     #    echo "PROGRESS $progress" > /mnt/.psplash/psplash_fifo
35     #fi
36 }
37
38 #
39 # Start script or program.
40 #
41 startup() {
42     # Handle verbosity
43     [ "$VERBOSE" = very ] && echo "INIT: Running $@"
44
45     case "$1" in
46         *.sh)
47             # Source shell script for speed.
48             (
49                 trap - INT QUIT TSTP
50                 scriptname=$1
51                 shift
52                 . $scriptname
53             )
54             ;;
55         *)
56             "$@"
57             ;;
58     esac
59     startup_progress
60 }
61
62 # Ignore CTRL-C only in this shell, so we can interrupt subprocesses.
63 trap ":" INT QUIT TSTP
64
65 # Set onlcr to avoid staircase effect.

```

```

66  stty onlcr 0>&1
67
68  # Limit stack size for startup scripts
69  [ "$STACK_SIZE" == "" ] || ulimit -S -s $STACK_SIZE
70
71  # Now find out what the current and what the previous runlevel are.
72
73  runlevel=$RUNLEVEL
74  # Get first argument. Set new runlevel to this argument.
75  [ "$1" != "" ] && runlevel=$1
76  if [ "$runlevel" = "" ]
77  then
78      echo "Usage: $0 <runlevel>" >&2
79      exit 1
80  fi
81  previous=$PREVLEVEL
82  [ "$previous" = "" ] && previous=N
83
84  export runlevel previous
85
86  # Is there an rc directory for this new runlevel?
87  if [ -d /etc/rc$runlevel.d ]
88  then
89      # Find out where in the progress bar the initramfs got to.
90      PROGRESS_STATE=0
91      if [ -f /dev/.initramfs/progress_state ]; then
92          # . /dev/.initramfs/progress_state
93      fi
94
95      # Split the remaining portion of the progress bar into thirds
96      progress_size=$((100 - $PROGRESS_STATE) / 3))
97
98      case "$runlevel" in
99          0|6)
100              # Count down from -100 to 0 and use the entire bar
101              first_step=-100
102              progress_size=100
103              step_change=1
104              ;;
105              S)
106              # Begin where the initramfs left off and use 2/3
107              # of the remaining space
108              first_step=$PROGRESS_STATE
109              progress_size=$((progress_size * 2))
110              step_change=1
111              ;;
112              *)
113              # Begin where rcS left off and use the final 1/3 of
114              # the space (by leaving progress_size unchanged)
115              first_step=$((progress_size * 2 + $PROGRESS_STATE))
116              step_change=1
117              ;;
118      esac
119
120      num_steps=0
121      for s in /etc/rc$runlevel.d/[SK]*; do
122          case "${s##/etc/rc$runlevel.d/S?.}" in
123              gdm|xdm|kdm|reboot|halt)
124                  break
125              ;;
126          esac
127          num_steps=$((num_steps + 1))
128      done
129      step=0
130
131      # First, run the KILL scripts.

```

```

132     if [ $previous != N ]
133     then
134         for i in /etc/rc$runlevel.d/K[0-9][0-9]*
135         do
136             # Check if the script is there.
137             [ ! -f $i ] && continue
138
139             # Stop the service.
140             startup $i stop
141         done
142     fi
143
144     # Now run the START scripts for this runlevel.
145     for i in /etc/rc$runlevel.d/S*
146     do
147         [ ! -f $i ] && continue
148
149         if [ $previous != N ] && [ $previous != S ]
150         then
151             #
152             # Find start script in previous runlevel and
153             # stop script in this runlevel.
154             #
155             suffix=${i#/etc/rc$runlevel.d/S[0-9][0-9]}
156             stop=/etc/rc$runlevel.d/K[0-9][0-9]$suffix
157             previous_start=/etc/rc$previous.d/S[0-9][0-9]$suffix
158             #
159             # If there is a start script in the previous level
160             # and _no_ stop script in this level, we don't
161             # have to re-start the service.
162             #
163             [ -f $previous_start ] && [ ! -f $stop ] && continue
164         fi
165         case "$runlevel" in
166             0|6)
167                 startup $i stop
168                 ;;
169             *)
170                 startup $i start
171                 ;;
172         esac
173     done
174 fi
175
176 #Uncomment to cause psplash to exit manually, otherwise it exits when it sees a VC switch
177 if [ "x$runlevel" != "xS" ] && [ ! -x /etc/rc${runlevel}.d/S??xserver-nodm ]; then
178     if type psplash-write >/dev/null 2>&1; then
179         TMPDIR=/mnt/.psplash psplash-write "QUIT" || true
180         umount -l /mnt/.psplash
181     fi
182 fi
183

```