Linux Magic System Request Key Hacks

Documentation for sysrq.c
Last update: 2007-JAN-06

What is the magic SysRq key?

It is a 'magical' key combo you can hit which the kernel will respond to
regardless of whatever else it is doing, unless it is completely locked up.

How do I use the magic SysRq key?

On x86   - You press the key combo 'ALT-SysRq-<command key>'. Note - Some
           keyboards may not have a key labeled 'SysRq'. The 'SysRq' key is
           also known as the 'Print Screen' key. Also some keyboards cannot
           handle so many keys being pressed at the same time, so you might
           have better luck with "press Alt", "press SysRq", "release Alt",
           "press <command key>", release everything.

On SPARC - You press 'ALT-STOP-<command key>', I believe.

On the serial console (PC style standard serial ports only) -
           You send a BREAK, then within 5 seconds a command key. Sending
           BREAK twice is interpreted as a normal BREAK.

On PowerPC - Press 'ALT - Print Screen (or F13) - <command key>,
             Print Screen (or F13) - <command key> may suffice.

On other - If you know of the key combos for other architectures, please
           let me know so I can add them to this section.

On all - write a character to /proc/sysrq-trigger. e.g.:

                echo t > /proc/sysrq-trigger

What are the 'command' keys?

'r'   - Turns off keyboard raw mode and sets it to XLATE.

'k'   - Secure Access Key (SAK) Kills all programs on the current virtual
          console. NOTE: See important comments below in SAK section.

'b'   - Will immediately reboot the system without syncing or unmounting
          your disks.

'c'   - Will perform a kexec reboot in order to take a crashdump.

'd'   - Shows all locks that are held.

'o'   - Will shut your system off (if configured and supported).

's'   - Will attempt to sync all mounted filesystems.

'u'   - Will attempt to remount all mounted filesystems read-only.

'p'   - Will dump the current registers and flags to your console.

't'   - Will dump a list of current tasks and their information to your
          console.

```
'm'     - Will dump current memory info to your console.

'n'     - Used to make RT tasks nice-able

'v'     - Dumps Voyager SMP processor info to your console.

'w'     - Dumps tasks that are in uninterruptable (blocked) state.

'x'     - Used by xmon interface on ppc/powerpc platforms.

'0'-'9' - Sets the console log level, controlling which kernel messages
          will be printed to your console. ('0', for example would make
          it so that only emergency messages like PANICs or OOPSes would
          make it to your console.)

'f'     - Will call oom_kill to kill a memory hog process.

'e'     - Send a SIGTERM to all processes, except for init.

'g'     - Used by kgdb on ppc platforms.

'i'     - Send a SIGKILL to all processes, except for init.

'h'     - Will display help (actually any other key than those listed
          above will display help. but 'h' is easy to remember :-)
```

Okay, so what can I use them for?

```
Well, re'B'oot is good when you're unable to shut down. But you should also
'S'ync and 'U'mount first. (The SUB-rule)

When a crash occurs, it is good practive to use the '1' command (one, not L)
followed by 'T'asks. If using a text console, you should see a trace dumped to
the screen. If the system is sufficiently alive, it will also be logged to
/var/log/kern.log and visible in the output from dmesg. This information shows
where the crash occurred, and should be included in any problem reports.

'C'rashdump can be used to manually trigger a complete crashdump when the system
is hung. The kernel needs to have been built with CONFIG_KEXEC enabled.

'S'ync is great when your system is locked up, it allows you to sync your
disks and will certainly lessen the chance of data loss and fscking. Note
that the sync hasn't taken place until you see the "OK" and "Done" appear
on the screen. (If the kernel is really in strife, you may not ever get the
OK or Done message...)

'U'mount is basically useful in the same ways as 'S'ync. I generally 'S'ync,
'U'mount, then re'B'oot when my system locks. It's saved me many a fsck.
Again, the unmount (remount read-only) hasn't taken place until you see the
"OK" and "Done" message appear on the screen.

un'R'aw is very handy when your X server or a svgalib program crashes.

sa'K' (Secure Access Key) is useful when you want to be sure there is no
trojan program running at console which could grab your password
when you would try to login. It will kill all programs on given console,
thus letting you make sure that the login prompt you see is actually
the one from init, not some trojan program.
IMPORTANT: In its true form it is not a true SAK like the one in a :IMPORTANT
IMPORTANT: c2 compliant system, and it should not be mistaken as   :IMPORTANT
IMPORTANT: such.                                                   :IMPORTANT
```

It seems others find it useful as (System Attention Key) which is useful when you want to exit a program that will not let you switch consoles. (For example, X or a svgalib program.)

The loglevels '0'-'9' are useful when your console is being flooded with kernel messages you do not want to see. Selecting '0' will prevent all but the most urgent kernel messages from reaching your console. (They will still be logged if syslogd/klogd are alive, though.)

t'E'rm and k'I'll are useful if you have some sort of runaway process you are unable to kill any other way, especially if it's spawning other processes.

Sometimes SysRq seems to get 'stuck' after using it, what can I do?

That happens to me, also. I've found that tapping shift, alt, and control on both sides of the keyboard, and hitting an invalid sysrq sequence again will fix the problem. (i.e., something like alt-sysrq-z). Switching to another virtual console (ALT+Fn) and then back again should also help.

I hit SysRq, but nothing seems to happen, what's wrong?

There are some keyboards that send different scancodes for SysRq than the pre-defined 0x54. So if SysRq doesn't work out of the box for a certain keyboard, run 'showkey -s' to find out the proper scancode sequence. Then use 'setkeycodes <sequence> 84' to define this sequence to the usual SysRq code (84 is decimal for 0x54). It's probably best to put this command in a boot script. Oh, and by the way, you exit 'showkey' by not typing anything for ten seconds.

Where can I get more information?

Check the latest official documentation at
http://lxr.linux.no/source/Documentation/sysrq.txt

Credits

Written by Mydraal <vulpyne[AT]vulpyne[DOT]net>
Updated by Adam Sulmicki <adam[AT]cfar.umd[DOT]edu>
Updated by Jeremy M. Dolan <jmd[AT]turbogeek[DOT]org> 2001/01/28 10:15:59
Added to by Crutcher Dunnavant <crutcher+kernel[AT]datastacks[DOT]com>
Formatted as PDF by Fredrik Wollsén <me[AT]motin[DOT]eu>

(This version includes some extra info from
https://help.ubuntu.com/community/DebuggingSystemCrash, and excludes ”How do I enable the magic SysRq key?”, and ”I want to add SysRQ key events to a module, how does it work?”. Refer to the complete official documentation for these sections)