```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
 <META NAME="GENERATOR" CONTENT="SGML-Tools 1.0.9">
 <TITLE>The Linux keyboard and console HOWTO: Unusual keys and keyboards</TITLE>
 <LINK HREF="kbd.FAQ-15.html" REL=next>
 <LINK HREF="kbd.FAQ-13.html" REL=previous>
 <LINK HREF="kbd.FAQ.html#toc14" REL=contents>
</HEAD>
<BODY>
<A HREF="kbd.FAQ-15.html">Next</A>
<A HREF="kbd.FAQ-13.html">Previous</A>
<A HREF="kbd.FAQ.html#toc14">Contents</A>
<HR>
<H2><A NAME="s14">14. Unusual keys and keyboards</A></H2>

<P>
<!--
keyboard!unusual versions of
-->

<!--
keyboard!non-standard keys on
-->
<P>The two keys PrintScrn/SysRq and Pause/Break are special in that they
have two keycodes: the former has keycode 84 when Alt is pressed
simultaneously, and keycode 99 otherwise; the latter has keycode
101 when Ctrl is pressed simultaneously, and keycode 119 otherwise.
(Thus, it makes no sense to bind functions to Alt keycode 99 or
Ctrl keycode 119.) The Pause/Break key is also special in another way:
it does not generate key-up scancodes, but generates the entire
6-scancode sequence on key-down.
<P>If you have strange keys, that do not generate any code under Linux
(or generate messages like "unrecognized scancode"), and your kernel
is 1.1.63 or later, then you can use setkeycodes(1) to tell the kernel
about them. Once they have gotten a keycode from <CODE>setkeycodes</CODE>,
they can be assigned a function by <CODE>loadkeys</CODE>.
<P>For example, using <CODE>showkey -s</CODE> one sees that Microsoft keyboards
use the scancode sequences (in hexadecimal) e0 5b (left Windows key),
e0 5c (right Windows key), e0 5d (Menu key).
Microsoft Internet keyboard also uses e0 6a (Back), e0 69 (Forward),
e0 68 (Stop), e0 6c (Mail), e0 65 (Search), e0 66 (Favorites),
e0 32 (Web/Home), e0 6b (My Computer), e0 21 (Calculator), e0 5f (Sleep).
Use <CODE>dumpkeys</CODE> to see what keycodes are still unused.
Typically values like 89-95 and 112-118 and 120-127 are free.
Now
<BLOCKQUOTE><CODE>
<PRE>
        % setkeycodes e05b 125
        % setkeycodes e05c 126
        % setkeycodes e05d 127
</PRE>
</CODE></BLOCKQUOTE>

assigns keycodes to these scancode sequences, and
<BLOCKQUOTE><CODE>
<PRE>
        % loadkeys
        keycode 125 = Decr_Console
        keycode 126 = Incr_Console
        keycode 127 = KeyboardSignal
        %
</PRE>
</CODE></BLOCKQUOTE>

would make these Windows keys go to the previous or next virtual console,
and let the Menu key create a fresh virtual console (in case you have
something like <CODE>spawn_console</CODE> running).
<P>
<H2><A NAME="ss14.1">14.1 Funkeys</A>
</H2>

<P>Many modern keyboards have buttons or keys with labels like
"Vol Up", "Eject" etc. that suggest actions rather than strings.
Of course one can bind shell commands to them, but then they’ll
work only when you are at a shell prompt.
Rick van Rein wrote a package funkey consisting of a kernel patch
and a daemon. The kernel patch creates a new character device,
and adds a new key type to indicate which keystrokes should be
sent to this new character device. A daemon can now listen to
the character device, somewhat like <CODE>gpm</CODE> listens to the
mouse device, and perform the actions indicated in its config file.
See
<A HREF="http://rick.vanrein.org/linux/funkey">rick.vanrein.org/linux/funkey</A>.
<P>
<HR>
<A HREF="kbd.FAQ-15.html">Next</A>
<A HREF="kbd.FAQ-13.html">Previous</A>
<A HREF="kbd.FAQ.html#toc14">Contents</A>
</BODY>
</HTML>
```