NAME

acpid − Advanced Configuration and Power Interface event daemon

SYNOPSIS

acpid [*options*]

DESCRIPTION

acpid is designed to notify user-space programs of ACPI events. **acpid** should be started during the system boot, and will run as a background process, by default. It will open an events file (*/proc/acpi/event* by default) and attempt to read whole lines which represent ACPI events. If the events file does not exist, **acpid** will attempt to connect to the Linux kernel via the input layer and netlink. When an ACPI event is received from one of these sources, **acpid** will examine a list of rules, and execute the rules that match the event. **acpid** will ignore all incoming ACPI events if a lock file exists (*/var/lock/acpid* by default).

Rules are defined by simple configuration files. **acpid** will look in a configuration directory (*/etc/acpi/events* by default), and parse all regular files with names that consist entirely of upper and lower case letters, digits, underscores, and hyphens (similar to **run-parts**(8)) that do not begin with a period ('.') or end with a tilde (~). Each file must define two things: an *event* and an *action*. Any blank lines, or lines where the first character is a hash ('#') are ignored. Extraneous lines are flagged as warnings, but are not fatal. Each line has three tokens: the key, a literal equal sign, and the value. The key can be up to 63 characters, and is case-insensitive (but whitespace matters). The value can be up to 511 characters, and is case and whitespace sensitive.

The event value is a regular expression (see **regcomp**(3)), against which events are matched.

The action value is a commandline, which will be invoked via */bin/sh* whenever an event matching the rule in question occurs. The commandline may include shell-special characters, and they will be preserved. The only special characters in an action value are "%" escaped. The string "%e" will be replaced by the literal text of the event for which the action was invoked. This string may contain spaces, so the commandline must take care to quote the "%e" if it wants a single token. The string "%%" will be replaced by a literal "%". All other "%" escapes are reserved, and will cause a rule to not load.

This feature allows multiple rules to be defined for the same event (though no ordering is guaranteed), as well as one rule to be defined for multiple events. To force **acpid** to reload the rule configuration, send it a SIGHUP.

The pseudo-action *<drop>* causes the event to be dropped completely and no further processing undertaken; clients connecting via the UNIX domain socket (see below) will not be notified of the event. This may be useful on some machines, such as certain laptops which generate spurious battery events at frequent intervals. The name of this pseudo-action may be redefined with a commandline option.

In addition to rule files, **acpid** also accepts connections on a UNIX domain socket (*/var/run/acpid.socket* by default). Any application may connect to this socket. Once connected, **acpid** will send the text of all ACPI events to the client. The client has the responsibility of filtering for messages about which it cares. **acpid** will not close the client socket except in the case of a SIGHUP or **acpid** exiting.

For faster startup, this socket can be passed in as stdin so that **acpid** need not create the socket. In addition, if a socket is passed in as stdin, **acpid** will not daemonize. It will be run in foreground. This behavior is provided to support **systemd**(1).

acpid will log all of its activities, as well as the stdout and stderr of any actions, to syslog.

All the default files and directories can be changed with commandline options.

OPTIONS

−c, **−−confdir** *directory*

This option changes the directory in which **acpid** looks for rule configuration files. Default is */etc/acpi/events*.

−C, **−−clientmax** *number*
This option changes the maximum number of non-root socket connections which can be made to the **acpid** socket. Default is *256*.

−d, **−−debug** This option increases the **acpid** debug level by one.

−e, **−−eventfile** *filename*
This option changes the event file from which **acpid** reads events. Default is */proc/acpi/event*.

−n, **−−netlink**
This option forces **acpid** to use the Linux kernel input layer and netlink interface for ACPI events.

−f, **−−foreground**
This option keeps **acpid** in the foreground by not forking at startup, and makes it log to stderr instead of syslog.

−l, **−−logevents**
This option tells **acpid** to log information about all events and actions.

−L, **−−lockfile** *filename*
This option changes the lock file used to stop event processing. Default is */var/lock/acpid*.

−g, **−−socketgroup** *groupname*
This option changes the group ownership of the UNIX domain socket to which **acpid** publishes events.

−m, **−−socketmode** *mode*
This option changes the permissions of the UNIX domain socket to which **acpid** publishes events. Default is *0666*.

−s, **−−socketfile** *filename*
This option changes the name of the UNIX domain socket which **acpid** opens. Default is */var/run/acpid.socket*.

−S, **−−nosocket**
This option tells **acpid** not to open a UNIX domain socket. This overrides the *-s* option, and negates all other socket options.

−p, **−−pidfile** *filename*
This option tells **acpid** to use the specified file as its pidfile. If the file exists, it will be removed and over-written. Default is */var/run/acpid.pid*.

−r, **−−dropaction** *action*
This option defines the pseudo-action which tells **acpid** to abort all processing of an event, including client notifications. Default is *<drop>*.

−t, **−−tpmutefix**
This option enables special handling of the mute button for certain ThinkPad models with mute LEDs that get out of sync with the mute state when the mute button is held down. With this option, the mute button will generate the following events in sync with the number of presses (and, by extension, the state of the LED):

button/mute MUTE (key pressed) K
button/mute MUTE (key released) K

−v, **−−version**
Print version information and exit.

−h, **−−help** Show help and exit.

EXAMPLE

This example will shut down your system if you press the power button.

Create a file named /etc/acpi/events/power that contains the following:

```
event=button/power
action=/etc/acpi/power.sh "%e"
```

Then create a file named /etc/acpi/power.sh that contains the following:

```
/sbin/shutdown -h now "Power button pressed"
```

Now, when **acpid** is running, a press of the power button will cause the rule in /etc/acpi/events/power to trigger the script in /etc/acpi/power.sh. The script will then shut down the system.

TROUBLESHOOTING

acpid is a simple program that runs scripts in response to ACPI events from the kernel. When there's trouble, the problem is rarely with **acpid** itself. The following are some suggestions for finding the most common sources of ACPI-related problems.

When troubleshooting **acpid**, it is important to be aware that other parts of a system might be handling ACPI events. **systemd**(1) is capable of handling the power switch and various other events that are commonly handled by **acpid**. See the description of HandlePowerKey in **logind.conf**(5) for more. Some window managers also take over **acpid**'s normal handling of the power button and other events.

kacpimon(8) can be used to verify that the expected ACPI events are coming in. See the man page for **kacpimon**(8) for the proper procedure. If the events aren't coming in, you've probably got a kernel driver issue.

If the expected events are coming in, then you'll need to check and see if your window manager is responsible for handling these events. Some are, some aren't. (E.g. in Ubuntu 14.04 (Unity/GNOME), there are settings for the laptop lid in the System Settings > Power > "When the lid is closed" fields.) If your window manager is responsible for handling the problematic event, and you've got it configured properly, then you may have a window manager issue.

Lastly, take a look in */etc/acpi/events* (see above). Is there a configuration file in there for the event in question (e.g. /etc/acpi/events/lidbtn for laptop lid open/close events)? Is it properly connected to a script (e.g. /etc/acpi/lid.sh)? Is that script working? It's not unusual for an **acpid** script to check and see if there is a window manager running, then do nothing if there is. This means it is up to the window manager to handle this event.

DEPENDENCIES

acpid should work on any linux kernel released since 2003.

FILES

/proc/acpi/event
/dev/input/event*
/etc/acpi/
/var/run/acpid.socket
/var/run/acpid.pid
/var/lock/acpid

BUGS

There are no known bugs. To file bug reports, see **PROJECT WEBSITE** below.

SEE ALSO

regcomp(3) **sh**(1) **socket**(2) **connect**(2) **init**(1) **systemd**(1) **acpi_listen**(8) **kacpimon**(8)

PROJECT WEBSITE

http://sourceforge.net/projects/acpid2/

AUTHORS

Ted Felix <ted@tedfelix.com>
Tim Hockin <thockin@hockin.org>
Andrew Henroid