Ways to Use the include Directive

Contents

1 Introduction
2 Preparation
3 Targets
  3.1 Reliable Failover
  3.2 Easy Switchover
  3.3 Advanced and Easy Switchover
4 Email Notifications
  4.1 Pager Duty
  4.2 Failure Levels
5 Times, Days, and Dates

Introduction

The include directive provides an easy and powerful way to reuse settings that are common to many events. What follows are some simple ways to leverage the include directive with a number of benefits:

- Reduce boilerplate settings.
- Make wholesale updates trivial.
- Add clarity to certain kinds of events.

Preparation

For the purposes of this article, many of the includeable events will be anchored under in the events tree at `includes/`:

```
~/.hcron/
    <hcronfqdn>/
        events/
            includes/
                ...
```

Then, to include in an event file:

```
...
include /includes/...
...
```

Targets

Reliable Failover

One of the most common uses of failover events is to send notifications. But, if there are problems going on, perhaps across multiple hosts, it is critical that the host on which the failover event is run is reliable. If a single failover event is being used, then the `host` setting can be set to the reliable host. But, if multiple failover events are being used, it may be preferable to have a basic failover event be defined which can be included by the others:

/includes/failover-host

```
host=rocksolid.abc.xyz
```

This can then be used by the various failover events:

/failover/server-not-answering

```
include /includes/failover-host
...
```

An alternative would be to use a variable in an event file:

/includes/hosts

```
FAILOVER_HOST=rocksolid.abc.xyz
```

and include it and reference the variable:

```
include /includes/hosts
host=$FAILOVER_HOST
```

Easy Switchover

In an environment running parallel systems: development and production, there are times when the production system becomes unavailable because of planned maintenance or unplanned events such as power failure or hardware failures. Being able to deal with such situations easily and quickly may be required to meet critical needs.

One way to handle this is to encode a host name in the event name. For example, if an event is defined at `/hosta/suitea`, then `hosta` can be used as:

/hosta/suitea

```
host=$HCRON_EVENT_NAME[0]
```

and when/if a switchover is needed, a simple rename and reload would be done:

```
mv hosta hostb
hcron-reload
```

Alternatively, the include directive can be used. Starting with an includeable event:

/includes/runhost

```
host=hosta
```

it can be included as:

```
include /includes/runhost
```

and a switchover be done by changing the `/includes/runhost` event and a reload:

```
vi runhost
hcron-reload
```

Each method has its pros and cons. The main pro for using an include file is that the event tree does not need to be changed, only the include file.

Advanced and Easy Switchover

Unlike the event name method, the include file method can be extended to handle many hosts by using variable names:

/includes/runhosts

```
SUITEA_RUNHOST=hosta0
SUITEB_RUNHOST=hosta1
SUITEC_RUNHOST=hosta2
```

and for the various suite event files:

/suitea/run0

```
include /includes/runhosts
host=$SUITEA_RUNHOST
```

/suiteb/run0

```
include /includes/runhosts
host=$SUITEB_RUNHOST
```

/suitec/run0

```
include /includes/runhosts
host=$SUITEC_RUNHOST
```

Then the settings in the `/includes/runhosts` event can be updated as needed to deal with maintenance, outages, or relocation of systems:

```
vi runhosts
hcron-reload
```

Variations and combinations of the above are also possible.

Email Notifications

In an enterprise context, it is not unusual for email to be sent as notification when an event fails. If there are a lot of events, then the recipient must be specified in each. As well, there may be different recipients for different kinds of failures. This can be easily handled using include events.

Pager Duty

To handle situations that need the attention of the person on pager a dedicated pager email is often set up. An alternative is to set up a includeable event snippet with the setting:

/includes/email-pager

```
notify_email=bill@abc.xyz
notify_message=paged for $HCRON_EVENT_NAME
```

Failure Levels

Suppose a suite of events can fail with a variety of severity levels: warning, serious, emergency.

/includes/email-warning

```
notify_email=manager-a@abc.xyz
notify_message=WARNING: event $HCRON_EVENT_NAME
```

/includes/email-serious

```
notify_email=manager-b@abc.xyz
notify_message=SERIOUS: event $HCRON_EVENT_NAME
```

/includes/email-emergency

```
notify_email=manager-c@abc.xyz
notify_message=EMERGENCY: event $HCRON_EVENT_NAME
```

In each case, a different recipient is specified and a different message is specified.

Times, Days, and Dates

There are a handful of standard `when_*` settings that many events use:

- every minute
- every hour
- every day

These can be set up as:

/includes/minutely

```
when_month=*
when_day=*
when_hour=*
when_minute=*
when_dow=*
```

/includes/hourly

```
when_month=*
when_day=*
when_hour=*
when_minute=0
when_dow=*
```

/includes/daily

```
when_month=*
when_day=*
when_hour=0
when_minute=0
when_dow=*
```

A notification event using one of these:

/remindme

```
as_user=
host=abc.xyz
command=
notify_email=bigman@abc.xyz
notify_message=You're friendly reminder ...
include /includes/hourly
```

If one of the settings is not suitable, it can be overridden (e.g., at the 30m mark is preferred over the 0m mark) :

/remindme

```
as_user=
host=abc.xyz
command=
notify_email=bigman@abc.xyz
notify_message=You're friendly reminder ...
include /includes/hourly
when_minute=30
```