NAME

inittab – script for init

DESCRIPTION

The file **/etc/inittab** controls process dispatching by **init**. The processes most typically dispatched by **init** are daemons.

The **inittab** file is composed of entries that are position dependent and have the following format:

id:*rstate*:*action*:*process*

Each entry is delimited by a newline; however, a backslash (\) preceding a newline indicates a continuation of the entry. Up to 512 characters for each entry are permitted. Comments may be inserted in the *process* field using the convention for comments described in **sh**(1). There are no limits (other than maximum entry size) imposed on the number of entries in the **inittab** file. The entry fields are:

id One or two characters used to uniquely identify an entry.

rstate Define the run level in which this entry is to be processed. Run-levels effectively correspond to a configuration of processes in the system. That is, each process spawned by **init** is assigned a run level(s) in which it is allowed to exist. The run levels are represented by a number ranging from **0** through **6**. For example, if the system is in run level **1**, only those entries having a **1** in the *rstate* field are processed.

When **init** is requested to change run levels, all processes that do not have an entry in the *rstate* field for the target run level are sent the warning signal **SIGTERM** and allowed a 5-second grace period before being forcibly terminated by the kill signal **SIGKILL**. The *rstate* field can define multiple run levels for a process by selecting more than one run level in any combination from **0** through **6**. If no run level is specified, then the process is assumed to be valid at all run levels **0** through **6**.

There are three other values, **a**, **b** and **c**, which can appear in the *rstate* field, even though they are not true run levels. Entries which have these characters in the *rstate* field are processed only when an **init** or **telinit** process requests them to be run (regardless of the current run level of the system). See **init**(1M). These differ from run levels in that **init** can never enter run level **a**, **b** or **c**. Also, a request for the execution of any of these processes does not change the current run level. Furthermore, a process started by an **a**, **b** or **c** command is not killed when **init** changes levels. They are killed only if their line in **inittab** is marked **off** in the *action* field, their line is deleted entirely from **inittab**, or **init** goes into single-user state.

action Key words in this field tell **init** how to treat the process specified in the *process* field. The actions recognized by **init** are as follows:

respawn If the process does not exist, then start the process; do not wait for its termination (continue scanning the **inittab** file), and when the process dies, restart the process. If the process currently exists, do nothing and continue scanning the **inittab** file.

wait When **init** enters the run level that matches the entry's *rstate*, start the process and wait for its termination. All subsequent reads of the **inittab** file while **init** is in the same run level cause **init** to ignore this entry.

once When **init** enters a run level that matches the entry's *rstate*, start the process, do not wait for its termination. When it dies, do not restart the process. If **init** enters a new run level and the process is still running from a previous run level change, the program is not restarted.

boot The entry is to be processed only at **init**'s boot-time read of the **inittab** file. **init** is to start the process and not wait for its termination; when it dies, it does not restart the process. In order for this instruction to be meaningful, the *rstate* should be the default or it must match **init**'s run level at boot time. This action is useful for an initialization function following a hardware reboot of the system.

bootwait The entry is to be processed the first time **init** goes from single-user to multi-user state after the system is booted. (If **initdefault** is set to **2**, the process runs right after the boot.) **init** starts the process, waits for its termination and, when it dies, does not restart the process.

powerfail Execute the process associated with this entry only when **init** receives a power fail signal, **SIGPWR** (see **signal**(3C)).

powerwait Execute the process associated with this entry only when **init** receives a power fail signal, **SIGPWR**, and wait until it terminates before continuing any processing of **inittab**.

off If the process associated with this entry is currently running, send the warning signal **SIGTERM** and wait 5 seconds before forcibly terminating the process with the kill signal **SIGKILL**. If the process is nonexistent, ignore the entry.

ondemand This instruction is really a synonym for the **respawn** action. It is functionally identical to **respawn** but is given a different keyword in order to divorce its association with run levels. This instruction is used only with the **a**, **b** or **c** values described in the *rstate* field.

initdefault An entry with this action is scanned only when **init** is initially invoked. **init** uses this entry to determine which run level to enter initially. It does this by taking the highest run level specified in the *rstate* field and using that as its initial state. If the *rstate* field is empty, this is interpreted as **0123456** and **init** will enter run level **6**. This will cause the system to loop (it will go to firmware and reboot continuously). Additionally, if **init** does not find an **initdefault** entry in **inittab**, it requests an initial run level from the user at reboot time.

sysinit Entries of this type are executed before **init** tries to access the console (that is, before the **Console Login:** prompt). It is expected that this entry will be used only to initialize devices that **init** might try to ask the run level question. These entries are executed and **init** waits for their completion before continuing.

process Specify a command to be executed. The entire **process** field is prefixed with **exec** and passed to a forked **sh** as **sh −c ′exec** *command*′. For this reason, any legal **sh** syntax can appear in the *process* field.

SEE ALSO

sh(1), **who**(1), **init**(1M), **ttymon**(1M), **exec**(2), **open**(2), **signal**(3C)