Oracle® Solaris Studio 12.3: Distributed Make (dmake)

Part No: E25220
December 2011

ORACLE

120118@25097

Contents

Preface

This manual contains the same content as the `dmake`(1) manual page.

Supported Platforms

This Oracle Solaris Studio release supports platforms that use the SPARC family of processor architectures running the Oracle Solaris operating system, as well as platforms that use the x86 family of processor architectures running Oracle Solaris or specific Linux systems.

This document uses the following terms to cite differences between x86 platforms:

- "x86" refers to the larger family of 64-bit and 32-bit x86 compatible products.
- "x64" points out specific 64-bit x86 compatible CPUs.
- "32-bit x86" points out specific 32-bit information about x86 based systems.

Information specific to Linux systems refers only to supported Linux x86 platforms, while information specific to Oracle Solaris systems refers only to supported Oracle Solaris platforms on SPARC and x86 systems.

For a complete list of supported hardware platforms and operating system releases, see the Oracle Solaris Studio 12.3 Release Notes.

Oracle Solaris Studio Documentation

You can find complete documentation for Oracle Solaris Studio software as follows:

- Product documentation is located at the Oracle Solaris Studio documentation web site, including release notes, reference manuals, user guides, and tutorials.
- Online help for the Code Analyzer, the Performance Analyzer, the Thread Analyzer, dbxtool, DLight, and the IDE is available through the Help menu, as well as through the F1 key and Help buttons on many windows and dialog boxes, in these tools.
- Man pages for command-line tools describe a tool's command options.

Resources for Developers

Visit the Oracle Technical Network web site to find these resources for developers using Oracle Solaris Studio:

- Articles on programming techniques and best practices
- Links to complete documentation for recent releases of the software
- Information on support levels
- User discussion forums.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Description	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for shells that are included in the Oracle Solaris OS. Note that the default system prompt that is displayed in command examples varies, depending on the Oracle Solaris release.

TABLE P–2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	`$`
Bash shell, Korn shell, and Bourne shell for superuser	`#`
C shell	`machine_name%`
C shell for superuser	`machine_name#`

Distributed Make (dmake)

This manual contains the same content as the dmake(1) manual page.

Synopsis

```
dmake [-c dmake_rcfile] [-g dmake_group] [-j dmake_max_jobs] [-m serial |
parallel | distributed | grid] [-o dmake_odir]
```

Description

This man page supplements the Oracle Solaris make utility man page. The dmake utility will run under the Linux operating system, but the dmake options will be independent of any other make utility.

Distributed Make (dmake) parses your makefiles and determines which target can be built concurrently, and distributes the build of those targets over a number of hosts set by you.

If you use the standard make utility on the Oracle Solaris operating system, the transition to dmake requires little if any alteration to your makefiles. dmake is a superset of the Oracle Solaris make utility, which can be used both on Oracle Solaris and Linux platforms. With nested makes, if a top-level makefile calls "make", you need to use $(MAKE).

You execute dmake on a dmake *host* and distribute *jobs* to *build servers*.

You can also distribute jobs to the dmake host, in which case the dmake host also becomes a build server. dmake distributes jobs based on makefile targets that dmake determines (based on your makefiles) can be built concurrently.

You can use any machine as a build server as long as it meets the following requirements:

- From the dmake host (the machine you are using) you must be able to use rsh or ssh without being prompted for a password to remotely execute commands on the build server.

  The following example shows the result of using rsh or ssh with the which command as a test on a machine running the Oracle Solaris operating system:

  ```
  demo% rsh machine_name which dmake
  /bin/dmake

  demo% ssh machine_name which dmake
  /bin/dmake
  ```

  The rsh or ssh call must be clean, returning no additional output.
- The bin directory in which the dmake software is installed must be accessible from the build server. By default, dmake assumes that the logical path to the dmake executables on the build server is the same as the dmake host. This assumption can be overridden by specifying a path name as an attribute of the host entry in the runtime configuration file.

From the dmake host you can control which build servers are used and how many dmake jobs are allotted to each build server.

The number of dmake jobs that can run on a given build server can also be limited on that server.

Options

These options and the environment variables and makefile macros described later in this man page modify the same behavior.

Their order of precedence is defined as:

1. Command-line options
2. Makefile macros
3. Environment variables
4. dmake default

-c *dmake_rcfile*

Specifies an alternate runtime configuration file.

The default runtime configuration file is $(HOME)/.dmakerc.

-g *dmake_group*

Specifies the name of the build server group to which jobs are distributed.

You define server groups in the runtime configuration file.

The default server group is the first group in the runtime configuration file.

-j *dmake_max_jobs*

Specifies the maximum total number of jobs that are distributed to the specified group of build servers in the runtime configuration file.

The default maximum number of jobs is the sum of all the specified jobs in a build server group.

The jobs are subtracted from, or added to, hosts by 1 in the order they appear in the runtime configuration file.

For example, if all jobs specified in the runtime configuration file total 8:

```
host earth { jobs = 3 }
host mars  { jobs = 5 }
```

and *dmake_max_jobs* is specified as 11, `dmake` adds three more jobs to the current total maximum number of jobs (which is eight) as follows:

```
host earth { jobs = 5 }
host mars  { jobs = 6 }
```

Also, if *dmake_max_jobs* is specified as 4, `dmake` subtracts four jobs (from the original eight) as follows:

```
host earth { jobs = 1 }
host mars  { jobs = 3 }
```

-m {serial | parallel | distributed | grid}

Specify one of the following key words:

- `serial`: Causes `dmake` to behave like the standard serial version of `make`.
- `parallel`: Causes `dmake` to distribute jobs to only the `dmake` host.
- `distributed`: Causes `dmake` to behave in fully distributed mode. This is the `dmake` default.
- `grid`: Causes `dmake` to use Oracle Grid Engine (formerly known as Solaris Grid Engine) to distribute build jobs.

`-o` *dmake_odir*

Specifies a common physical directory that `dmake` can write temporary output files to and read temporary output files from. The directory used is `$(HOME)/.dmake` and this or whichever directory is specified, must be visible to all build servers.

Use this option only if the `$(HOME)` directory on your local host and the `$(HOME)` directory on all of your remote hosts are not the same physical `$(HOME)` directory. For example, a root user would use this option.

Usage

Special-purpose Targets

`dmake` allows targets to be built concurrently on a number of build servers. Concurrent processing can greatly reduce the time required to build a large system or project. `dmake` supplies the following special makefile targets for controlling concurrency and timing.

- `.NO_PARALLEL`

  Use this target to indicate which targets are to be processed serially.

- `.PARALLEL`

  Use this target to indicate which targets are to be processed in parallel.

- `.LOCAL`

  Use this target to indicate which targets are to be processed serially on the local host.

- `.WAIT`

  When you specify this target in a dependency list, `dmake` waits until the dependencies that precede it are finished before processing those that follow, even when processing is parallel.

Makefiles that you write using these targets remain compatible with the standard version of `make` distributed with Oracle Solaris. Standard `make` accepts these targets without error (and without action).

Controlling `dmake` Jobs

The distribution of `dmake` jobs is controlled in two ways:

1. A `dmake` user on a `dmake` host can specify the machines they want to use as build servers and the number of jobs they want to distribute to each build server.
2. The owner (a user that can alter the `/etc/opt/SPROdmake/dmake.conf` file) on a build server can control the maximum total number of `dmake` jobs that can be distributed to that build server.

The dmake Host

When dmake begins execution it searches for a runtime configuration file to know where to distribute jobs.

Generally, this file is located in your home directory on the dmake host and is named .dmakerc.

dmake searches for the runtime configuration file in the following locations and in the following order:

1. The path name you specify on the command line using the -c option
2. The path name you specify using the DMAKE_RCFILE makefile macro
3. The path name you specify using the DMAKE_RCFILE environment variable
4. In $(HOME)/.dmakerc

If a runtime configuration file is not found, dmake switches to parallel mode and distributes two jobs (the default) to the dmake host. You can change this using the -j option, or DMAKE_MAX_JOBS.

The runtime configuration file may contain a list of build servers and the number of jobs you want distributed to each build server.

The following is a sample of a simple runtime configuration file:

```
# My machine. This entry causes dmake to distribute to it
falcon { jobs = 1 }
hawk
eagle { jobs = 3 }
# Manager's machine. She's usually at meetings
heron { jobs = 4 }
avocet
```

The entries: falcon, hawk, eagle, heron, and avocet are listed as build servers.

You can specify the number of jobs you want distributed to each build server. The default number of jobs is two.

Any line that begins with the "#" character is interpreted as a comment.

Note – This list of build servers includes falcon which is also the dmake host. The dmake host can also be specified as a build server. If you do not include it in the runtime configuration file, no dmake jobs are distributed to it.

You can also construct groups of build servers in the runtime configuration file. This provides you with the flexibility of easily switching between different groups of build servers as circumstances warrant. For instance you may define a different group of build servers for builds under different operating systems, or define groups of build servers that have special software installed on them. The build servers must be all the same architecture and have the same OS installed.

The following runtime configuration file contains groups:

```
earth                   { jobs = 2 }
mars                    { jobs = 3 }

group lab1 {
            host falcon   { jobs = 3 }
            host hawk
            host eagle    { jobs = 3 }
}

group lab2 {
            host heron
            host avocet   { jobs = 3 }
            host stilt    { jobs = 2 }


group labs {
            group lab1
            group lab2
}

group sunos5.x {
            group labs
            host jupiter
            host venus    {     jobs = 2 }
            host pluto    {     jobs = 3 }
}
```

User Commands

Formal groups are specified by the `group` directive and lists of their constituents are delimited by braces (`{ }`).

Build servers that are constituents of groups are specified by the optional `host` directive.

Groups can be constituents of other groups.

Individual build servers can be listed in runtime configuration files that also contain groups of build servers. In this case `dmake` treats these build servers as constituents of the unnamed group.

`dmake` distributes jobs to a single group of hosts specified by the following list and in precedence from 1 to 4.

1. The group specified on the command-line as an argument to the `-g` option.
2. The group specified by the `DMAKE_GROUP` makefile macro.
3. The group specified by the `DMAKE_GROUP` environment variable.
4. The first formal group listed in the runtime configuration file.

The names of groups and hosts specified in the runtime configuration file may be enclosed in double quotes. This is to allow more flexibility with respect to the character sequences that may appear as part of the group and host names. For example, if the name of the group starts with a digit it should be double-quoted:

```
group "123_sparc"
```

As mentioned above, the `bin` directory in which the `dmake` software is installed must be accessible from the build server. By default, `dmake` assumes that the logical path to the `dmake` executables on the build server is the same as the `dmake` host. This assumption can be overridden by specifying a path name as an attribute of the host entry in the runtime configuration file. For example:

```
group sparc-cluster {
   host wren   { jobs = 10 , path = "/export/solstudio/bin" }
   host stimpy { path = "/bin"                }
}
```

The Build Server

The `/etc/opt/SPROdmake/dmake.conf` file is located in the file system of build servers.

Use this file to specify the following:

- Required: The maximum total number of `dmake` jobs (from all users) that can run concurrently on that build server.
- Optional: The priority under which all `dmake` jobs are to be run.

The following is a sample of a `dmake.conf` file:

```
max_jobs: 8
nice_prio: 5
```

This file sets the maximum number of `dmake` jobs permitted to run on that build server (from all `dmake` users) to be eight. You can change the priority of the jobs to be run by using the `nice_prio` command. See `nice(1)`.

Note – If the `/etc/opt/SPROdmake/dmake.conf` file does not exist on a build server, no `dmake` jobs will be allowed to run on that server.

Specifying Which Remote Shell To Use

The path to the remote shell can be specified in the `.dmakerc` file.

For example:

```
host earth { jobs = 3 }
host mars  { jobs = 5 , rsh = "/bin/ssh" }
```

If `rsh =` is not specified, `dmake` will use `/bin/rsh` by default.

As with `rsh`, you must ensure that `ssh` can login to the remote host without requiring a password, and does not issue any warnings or errors.

Environment/Macros

The following can be defined as either environment variables or makefile macros:

DMAKE_RCFILE

Defines an alternate runtime configuration file. The default runtime configuration file is `$(HOME)/.dmakerc`.

DMAKE_GROUP

Defines the name of the build server group to which jobs are distributed. Server groups are defined in the runtime configuration file. The default server group is the first group in the runtime configuration file.

DMAKE_MAX_JOBS

Defines the maximum total number of jobs that are distributed to the specified group of build servers in the runtime configuration file. The default maximum number of jobs is the sum of all the specified jobs in a build server group. The jobs are subtracted from, or added to, hosts by 1 in the order they appear in the runtime configuration file.

See the `-j` option in this man page for an example.

DMAKE_ADJUST_MAX_JOBS

May contain one of the following key words:

- `YES`: Allows `dmake` to adjust the limit of parallel jobs according to the current loading of the system. If the system is not overloaded, `dmake` will use the limit defined by the user. If the system is overloaded, `dmake` will set the "current" limit to less than the limit defined by the user.

  If this variable is not set, `dmake` will adjust the limit of parallel jobs according to the current loading of the system. This is the `dmake` default.
- `NO`: Causes `dmake` to switch off the autoadjustment mechanism.

DMAKE_MODE

May contain one of the following key words:

- `serial`: Causes `dmake` to behave like the standard serial version of `make`.
- `parallel`: Causes `dmake` to distribute jobs to only the `dmake` host.
- `distributed`: Causes `dmake` to behave in fully distributed mode. This is the `dmake` default.
- `grid`: Causes `dmake` to use Oracle Grid Engine to distribute build jobs.

DMAKE_ODIR

Defines a common physical directory that `dmake` can write temporary output files to and read temporary output files from.

Use this environment variable, or macro, only if the `$(HOME)` directory on your local host and the `$(HOME)` directory on all of your remote hosts are not the same physical `$(HOME)` directory. For example, a root user would use this option.

DMAKE_OUTPUT_MODE

Defines the format of the log file. May contain one of the following keywords:

- `TEXT1`: On starting each build job, `dmake` prints the name of the system and command to the log file.

  Also, if the command itself prints any output, then as the job finishes, `dmake` prints the name of the system and command to the log file again along with the command output.

  Example:

  ```
  host1 --> 1 job
  echo "Done host1"
  host2 --> 1 job
  echo
  "Done host2"
  host1 --> Job output
  echo "Done host1"
  Done host1
  host2 --> Job output
  echo "Done host2"
  Done host2
  ```

  This is the `dmake` default.

- `TEXT2`: Allows `dmake` to serialize the output of parallel jobs, which makes the log file more readable. In this mode, `dmake` just prints the command once, immediately followed by the command output, as each job finishes.

  Example:

```
echo "Done host1"
Done host1
echo "Done host2"
Done host2
```

Files

- `$(HOME)/.dmakerc`: The default runtime configuration file. Contains the names of build servers and groups of build servers.
- `/etc/opt/SPROdmake/dmake.conf`: Located on build servers, this file is used to specify the maximum total number of jobs that can be distributed to it by all `dmake` users. It is also used to specify the priority all `dmake` jobs are to be run under.

See Also

Access the complete Oracle Solaris Studio documentation on the web at `http://www.oracle.com/technetwork/server-storage/solarisstudio/documentation/index.html`.

Consult the following man pages for additional information.

- `make`(1)
- `rsh`(1)
- `ssh`(1)
- `hosts`(4)
- `hosts.equiv`(4)
- `attributes`(5)
- `largefile`(5)