Distributed Make (`dmake`)

Sun™ Studio 11

Sun Microsystems, Inc.
www.sun.com

Part No. 819-3697-10
November 2005, Revision A

Submit comments about this document at: http://www.sun.com/hwdocs/feedback

Contents

Before You Begin

Distributed Make (`dmake`) describes the Sun Studio 11 `dmake` command. `dmake` parses your makefiles and determines the targets that can be built concurrently, and distributes the build of those targets over a number of hosts set by you.

The information in this document supplements the Solaris make utility man page.

Typographic Conventions

TABLE P-1 Typeface Conventions

Typeface	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123`	What you type, when contrasted with on-screen computer output	`%` **`su`** `Password:`
AaBbCc123	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this.
AaBbCc123	Command-line placeholder text; replace with a real name or value	To delete a file, type **`rm`** *filename*.

TABLE P-2 Code Conventions

Code Symbol	Meaning	Notation	Code Example
[ ]	Brackets contain arguments that are optional.	`O[`*n*`]`	`O4, O`
{ }	Braces contain a set of choices for a required option.	`d{y|n}`	`dy`
|	The "pipe" or "bar" symbol separates arguments, only one of which may be chosen.	`B{dynamic|static}`	`Bstatic`
:	The colon, like the comma, is sometimes used to separate arguments.	`R`*dir*`[:`*dir*`]`	`R/local/libs:/U/a`
…	The ellipsis indicates omission in a series.	`xinline=`*f1*`[,…`*fn*`]`	`xinline=alpha,dos`

Shell Prompts

Shell	Prompt
C shell	*machine-name*`%`
C shell superuser	*machine-name*`#`
Bourne shell and Korn shell	`$`
Superuser for Bourne shell and Korn shell	`#`

Supported Platforms

This Sun Studio release supports systems that use the SPARC® and x86 families of processor architectures: UltraSPARC®, SPARC64, AMD64, Pentium, and Xeon EM64T. The supported systems for the version of the Solaris Operating System you

are running are available in the hardware compatibility lists at http://www.sun.com/bigadmin/hcl. These documents cite any implementation differences between the platform types.

In this document, these x86 related terms mean the following:

- "x86" refers to the larger family of 64-bit and 32-bit x86 compatible products.
- "x64' points out specific 64-bit information about AMD64 or EM64T systems.
- "32-bit x86" points out specific 32-bit information about x86 based systems.

For supported systems, see the hardware compatibility lists.

Accessing Sun Studio Software and Man Pages

The compilers and tools and their man pages are not installed into the standard `/usr/bin/` and `/usr/share/man` directories. To access the compilers and tools, you must have your `PATH` environment variable set correctly (see "Accessing the Software" on page 7). To access the man pages, you must have the your `MANPATH` environment variable set correctly (see "Accessing the Man Pages" on page 8.).

For more information about the `PATH` variable, see the `csh`(1), `sh`(1), `ksh`(1, and `bash(1)` man pages. For more information about the `MANPATH` variable, see the `man`(1) man page. For more information about setting your `PATH` variable and `MANPATH` variable to access this release, see the installation guide or your system administrator.

Note – The information in this section assumes that your Sun Studio software is installed in the /opt directory on Solaris platforms and in the `/opt/sun` directory on Linux platforms. If your software is not installed in the default directory, ask your system administrator for the equivalent path on your system.

Accessing the Software

Use the steps below to determine whether you need to change your `PATH` variable to access the compilers and tools.

To Determine Whether You Need to Set Your PATH Environment Variable

1. **Display the current value of the PATH variable by typing the following at a command prompt.**

```
% echo $PATH
```

2. **On Solaris platforms, review the output to find a string of paths that contain /opt/SUNWspro/bin/. On Linux platforms, review the output to find a string of paths that contain /opt/sun/sunstudio11/bin.**

   If you find the path, your PATH variable is already set to access the compilers and tools. If you do not find the path, set your PATH environment variable by following the instructions in the next procedure.

To Set Your PATH Environment Variable to Enable Access to the Compilers and Tools

- **On Solaris platforms, add the following to your PATH environment variable. If you have previously installed Forte Developer software, Sun ONE Studio software, or another release of Sun Studio software, add the following path before the paths to those installations.**

  /opt/SUNWspro/bin

- **On Linux platforms, add the following path to your PATH environment variable:**

  /opt/sun/sunstudio11/bin

Accessing the Man Pages

Use the following steps to determine whether you need to change your MANPATH variable to access the man pages.

To Determine Whether You Need to Set Your MANPATH Environment Variable

1. **Request the dbx man page by typing the following at a command prompt.**

```
% man dbx
```

2. **Review the output, if any.**

   If the `dbx`(1) man page cannot be found or if the man page displayed is not for the current version of the software, follow the instructions in the next procedure to set your `MANPATH` environment variable.

To Set Your `MANPATH` Environment Variable to Enable Access to the Man Pages

- **On Solaris platforms, add the following path to your MANPATH environment variable:**

  `/opt/SUNWspro/man`

- **On Linux platforms, add the following path to your MANPATH environment variable:**

  `/opt/sun/sunstudio11/man`

Accessing the Integrated Development Environment

The Sun Studio integrated development environment (IDE) provides modules for creating, editing, building, debugging, and analyzing the performance of a C, C++, Java, or Fortran application.

The command to start the IDE is sunstudio. For details on this command, see the `sunstudio`(1) man page.

The correct operation of the IDE depends on the IDE being able to find the core platform. The sunstudio command looks for the core platform in two locations:

- The command looks first in the default installation directory, `/opt/netbeans/3.5V11` on Solaris platforms and `/opt/sun/netbeans/3.5V11` on Linux platforms.
- If the command does not find the core platform in the default directory, it assumes that the directory that contains the IDE and the directory that contains the core platform are both installed in or mounted to the same location. For example, on Solaris platforms, if the path to the directory that contains the IDE is `/foo/SUNWspro`, the command looks for the core platform in `/foo/netbeans/3.5V11` On Linux platforms, if the path to the directory that contains the IDE is `/foo/sunstudio11`, the command looks for the core platform in `/foo/netbeans/3.5V11`.

If the core platform is not installed or mounted to either of the locations where the sunstudio command looks for it, then each user on a client system must set the environment variable SPRO_NETBEANS_HOME to the location where the core platform is installed or mounted (*/installation_directory*`/netbeans/3.5V11`).

On Solaris platforms, each user of the IDE also must add */installation_directory*`/SUNWspro/bin` to theiR `$PATH` in front of the path to any other release of Forte Developer software, Sun ONE Studio software, or Sun Studio software. On Linux platforms, each user of the IDE also must add */installation_directory*`/sunstudio11/bin` to their `$PATH` in front of the path to any other release of Sun Studio software.

The path */installation_directory*`/netbeans/3.511/bin` should not be added to the user's `$PATH`.

Accessing Sun Studio Documentation

You can access the documentation at the following locations:

- The documentation is available from the documentation index that is installed with the software on your local system or network at `file:/opt/SUNWspro/docs/index.html` on Solaris platforms, and at `file:/opt/sun/sunstudio11/docs/index.html` on Linux platforms.

  If your software is not installed in the `/opt` directory on a Solaris platform or the `/opt/sun` directory on a Linux platform, ask your system administrator for the equivalent path on your system.
- Most manuals are available from the `docs.sun.com`sm web site. The following titles are available through your installed software only:
  - *Standard C++ Library Class Reference*
  - *Standard C++ Library User's Guide*
  - *Tools.h++ Class Library Reference*
  - *Tools.h++ User's Guide*
- The release notes are available from the `docs.sun.com` web site.
- Online help for all components of the IDE is available through the Help menu, as well as through Help buttons on many windows and dialog boxes, in the IDE.

The `docs.sun.com` web site (`http://docs.sun.com`) enables you to read, print, and buy Sun Microsystems manuals through the Internet. If you cannot find a manual, see the documentation index that is installed with the software on your local system or network.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with use of or reliance on any such content, goods, or services available on or through any such sites or resources.

Documentation in Accessible Formats

The documentation is provided in accessible formats that are readable by assistive technologies for users with disabilities. You can find accessible versions of documentation as described in the following table. If your software is not installed in the `/opt` directory, ask your system administrator for the equivalent path on your system.

Type of Documentation	Format and Location of Accessible Version
Manuals (except third-party manuals)	HTML at `http://docs.sun.com`
Third-party manuals: • *Standard C++ Library Class Reference* • *Standard C++ Library User's Guide* • *Tools.h++ Class Library Reference* • *Tools.h++ User's Guide*	HTML in the installed software through the documentation index at `file:/opt/SUNWspro/docs/index.html`
Readmes	HTML on the Sun Developer Network portal at `http://developers.sun.com/prodtech/cc/documentation/`
Man pages	HTML in the installed software through the documentation index at `file:/opt/SUNWspro/docs/index.html` on Solaris platforms, and at `file:/opt/sun/sunstudio11/docs/index.html` on Linux platforms.
Online help	HTML available through the Help menu and Help buttons in the IDE.
Release notes	HTML at `http://docs.sun.com`

Related Compilers and Tools Documentation

The following table describes related documentation that is available at `file:/opt/SUNWspro/docs/index.html` and `http://docs.sun.com`. If your software is not installed in the `/opt` directory, ask your system administrator for the equivalent path on your system.

Document Title	Description
Numerical Computation Guide	Describes issues regarding the numerical accuracy of floating-point computations.

Accessing Related Solaris Documentation

The following table describes related documentation that is available through the `docs.sun.com` web site.

Document Collection	Document Title	Description
Solaris Reference Manual Collection	See the titles of man page sections.	Provides information about the Solaris operating environment.
Solaris Software Developer Collection	*Linker and Libraries Guide*	Describes the operations of the Solaris link-editor and runtime linker.
Solaris Software Developer Collection	*Multithreaded Programming Guide*	Covers the POSIX and Solaris threads APIs, programming with synchronization objects, compiling multithreaded programs, and finding tools for multithreaded programs.

Resources for Developers

Visit the Sun Developer Network Sun Studio portal at http://developers.sun.com/prodtech/cc to find these frequently updated resources:

- Articles on programming techniques and best practices
- A knowledge base of short programming tips
- Documentation of compilers and tools components, as well as corrections to the documentation that is installed with your software
- Information on support levels
- User forums
- Downloadable code samples
- New technology previews

The Sun Studio portal is one of a number of additional resources for developers at the Sun Developer Network website, http://developers.sun.com.

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in this document, go to:

http://www.sun.com/service/contacting

Sending Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. Submit your comments to Sun at this URL

http://www.sun.com/hwdocs/feedback

Please include the part number (819-3697-10) of your document.

Distributed Make (dmake)

Synopsis

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

Description

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 have use the standard make utility on the Solaris Operating System (Solaris OS), the transition to dmake requires little if any alteration to your makefiles. dmake is a superset of the Solaris make utility, which can be used both on 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, without being prompted for a password, to remotely execute commands on the build server.

For example:

```
demo% rsh <machine_name> which dmake
/opt/SUNWspro/bin/dmake
```

The `rsh` must be "clean", with 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 a 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

`-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 4 jobs (from the original eight) as follows:

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

-m {serial | parallel | distributed}

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.

-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.

Note – Note: 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:

- Command-line options
- Makefile macros
- Environment variables
- dmake default

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 special makefile targets for controlling concurrency and timing.

- `.PARALLEL`
- `.NO_PARALLEL`
- `.LOCAL`
- `.WAIT`

`.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 Solaris 1.x and Solaris 2.x. 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 on groups of build servers that have special software installed on them. The build servers must be all the same architecture and have the same SunOS 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/SUNWspro/bin" }
   host stimpy { path = "/opt/SUNWspro/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 /usr/bin/ 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.

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.

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 `/usr/bin/nice` priority all `dmake` jobs are to be run under.

See Also

Access the complete Sun Studio 11 documentation set by pointing an HTML browser at `/opt/SUNWspro/docs/index.html` on Solaris platforms or `/opt/sun/sunstudio11/docs/index.html` on Linux platforms.

If the Sun Studio software is not installed in the default `/opt` directory, ask your system administrator for the equivalent path on your system.

Consult the following man pages for additional information.

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