NAME

imake – C preprocessor interface to the make utility

SYNOPSIS

imake [**-D***define*] [**-I***dir*] [**-T***template*] [**-f** *filename*] [**-s** *filename*] [**-e**] [**-v**]

DESCRIPTION

imake is used to generate **Makefiles** from a template, a set of **cpp**(1) macro functions, and a per-directory input file called an **Imakefile**. This allows machine dependencies (such has compiler options, alternate command names, and special **make**(1S) rules) to be kept separate from the descriptions of the various items to be built.

OPTIONS

The following command line options may be passed to **imake**:

–**D***define*
: This option is passed directly to **cpp**. It is typically used to set directory-specific variables. For example, the X Window System uses this flag to set *TOPDIR* to the name of the directory containing the top of the core distribution and *CURDIR* to the name of the current directory, relative to the top.

–**I***directory*
: This option is passed directly to **cpp**(1). It is typically used to indicate the directory in which the **imake** template and configuration files may be found.

–**T***template*
: This option specifies the name of the master template file (which is usually located in the directory specified with *–I*) used by **cpp**. The default is **Imake.tmpl**.

–**f** *filename*
: This option specifies the name of the per-directory input file. The default is **Imakefile**.

–**s** *filename*
: This option specifies the name of the **make** description file to be generated but **make** should not be invoked. If the *filename* is a dash (-), the output is written to **stdout**. The default is to generate, but not execute, a **Makefile**.

–**e**
: This option indicates the **imake** should execute the generated **Makefile**. The default is to leave this to the user.

–**v**
: This option indicates that **imake** should print the **cpp** command line that it is using to generate the **Makefile**.

HOW IT WORKS

imake invokes **cpp** with any –**I** or **-D** flags passed on the command line and passes it the following 3 lines:

```
#define IMAKE_TEMPLATE "Imake.tmpl"
#define INCLUDE_IMAKEFILE "Imakefile"
#include IMAKE_TEMPLATE
```

where *Imake.tmpl* and *Imakefile* may be overridden by the –**T** and –**f** command options, respectively. If the **Imakefile** contains any lines beginning with a '#' character that is not followed by a **cpp** directive (**#include**, **#define**, **#undef**, **#ifdef**, **#else**, **#endif**, or **#if**), **imake** will make a temporary **makefile** in which the '#' lines are prepended with the string ''/**/'' (so that **cpp** will copy the line into the **Makefile** as a comment).

The **Imakefile** reads in file containing machine-dependent parameters (specified as **cpp** symbols), a site-specific parameters file, a file containing **cpp** macro functions for generating **make** rules, and finally the **Imakefile** (specified by INCLUDE_IMAKEFILE) in the current directory. The **Imakefile** uses the macro functions to indicate what targets should be built; **imake** takes care of generating the appropriate rules.

The rules file (usually named **Imake.rules** in the configuration directory) contains a variety of **cpp** macro functions that are configured according to the current platform. **imake** replaces any occurrences of the string ''@@'' with a newline to allow macros that generate more than one line of **make** rules. For example, the macro

```
#define     program_target(program, objlist)                    @@\
program:    objlist                                             @@\
            $(CC) -o $@ objlist $(LDFLAGS)
```

when called with *program_target(foo, foo1.o foo2.o)* will expand to

```
foo:        foo1.o foo2.o
            $(CC) -o $@ foo1.o foo2.o $(LDFLAGS)
```

On systems whose **cpp** reduces multiple tabs and spaces to a single space, **imake** attempts to put back any necessary tabs (**make** is very picky about the difference between tabs and spaces). For this reason, colons (:) in command lines must be preceded by a backslash (\).

USE WITH THE X WINDOW SYSTEM

The X Window System uses **imake** extensively, for both full builds within the source tree and external software. As mentioned above, two special variables, *TOPDIR* and *CURDIR* set to make referencing files using relative path names easier. For example, the following command is generated automatically to build the **Makefile** in the directory **lib/X/** (relative to the top of the sources):

```
% ../.././config/imake -I../.././config \
        -DTOPDIR=../.././. -DCURDIR=./lib/X
```

When building X programs outside the source tree, a special symbol **UseInstalled** is defined and *TOPDIR* and *CURDIR* are omitted. If the configuration files have been properly installed, the script **xmkmf**(1) may be used to specify the proper options:

```
% xmkmf
```

The command **make Makefiles** can then be used to generate **Makefiles** in any subdirectories.

FILES

usr/tmp/tmp-imake.*nnnnnn*	temporary input file for **cpp**
usr/tmp/tmp-make.*nnnnnn*	temporary input file for **make**
usr/ccs/lib/cpp	default C preprocessor

SEE ALSO

make(1S), **execvp**(2), **xmkmf**(1)
S. I. Feldman *Make – A Program for Maintaining Computer Programs*

ENVIRONMENT VARIABLES

The following environment variables may be set, however their use is not recommended as they introduce dependencies that are not readily apparent when **imake** is run:

IMAKEINCLUDE
If defined, this should be a valid include argument for the C preprocessor. E.g. *-I* **/usr/include/local**. Actually, any valid **cpp** argument will work here.

IMAKECPP
If defined, this should be a valid path to a preprocessor program. E.g. **/usr/local/cpp**. By default, **imake** will use **/lib/cpp**.

IMAKEMAKE
If defined, this should be a valid path to a make program. E.g. **/usr/local/make**. By default, **imake** will use whatever **make** program is found using **execvp**(2).

BUGS

Comments should be preceded by ‘‘/**/#’’ to protect them from **cpp**.

AUTHOR

Todd Brunhoff, Tektronix and MIT Project Athena; Jim Fulton, MIT X Consortium