MAKEFILE - MACROS

The make program allows you to use macros, which are similar to variables. Macros are defined in a Makefile as = pairs. For example,

```
MACROS  = -me
PSROFF  = groff -Tps
DITROFF = groff -Tdvi
CFLAGS  = -O -systype bsd43
LIBS    = "-lncurses -lm -lsdl"
MYFACE  = ":*)"
```

Special Macros

Before issuing any command in a target rule set, there are certain special macros predefined −

- $@ is the name of the file to be made.
- $? is the names of the changed dependents.

For example, we could use a rule as follows

```
hello: main.cpp hello.cpp factorial.cpp
 $(CC) $(CFLAGS) $? $(LDFLAGS) -o $@

Alternatively –

hello: main.cpp hello.cpp factorial.cpp
 $(CC) $(CFLAGS) $@.cpp $(LDFLAGS) -o $@
```

In this example, action lines like *(CC)CFLAGS ?LDFLAGS -o $@* should start after a tab \t as shown above otherwise it will throw an error. Here $@ represents *hello* and ?*or*@.cpp picks up all the changed source files.

There are two more special macros used in the implicit rules. They are −

- $< the name of the related file that caused the action.
- $* the prefix shared by target and dependent files.

Common implicit rule is for the construction of .o *object* files out of .cpp *sourcefiles*.

```
.cpp.o:
 $(CC) $(CFLAGS) -c $<

Alternatively –

.cpp.o:
 $(CC) $(CFLAGS) -c $*.c
```

Conventional Macros

There are various default macros. You can see them by *typing " make − p " toprintoutthedefaults*. Most are pretty obvious from the rules in which they are used.

These predefined variables, i.e., macros used in implicit rules fall into two classes −

- Macros that are names of programs *suchasCC*
- Macros that contain arguments of the programs *suchasCFLAGS*.

Here is a table of some of the common variables used as names of programs in built-in rules of

makefiles.

AR	Archive-maintaining program; default is `ar'.
AS	Program for compiling assembly files; default is `as'.
CC	Program for compiling C programs; default is `cc'.
CO	Program for checking out files from RCS; default is `co'.
CXX	Program for compiling C++ programs; default is `g++'.
CPP	Program for running the C preprocessor, with results to standard output; default is `$(CC) -E'.
FC	Program for compiling or preprocessing Fortran and Ratfor programs; default is `f77'.
GET	Program for extracting a file from SCCS; default is `get'.
LEX	Program to use to turn Lex grammars into source code; default is `lex'.
YACC	Program to use to turn Yacc grammars into source code; default is `yacc'.
LINT	Program to use to run lint on source code; default is `lint'.
M2C	Program to use to compile Modula-2 source code; default is `m2c'.
PC	Program for compiling Pascal programs; default is `pc'.
MAKEINFO	Program to convert a Texinfo source file into an Info file; default is `makeinfo'.
TEX	Program to make TeX dvi files from TeX source; default is `tex'.
TEXI2DVI	Program to make TeX dvi files from Texinfo source; default is `texi2dvi'.
WEAVE	Program to translate Web into TeX; default is `weave'.
CWEAVE	Program to translate C Web into TeX; default is `cweave'.
TANGLE	Program to translate Web into Pascal; default is `tangle'.
CTANGLE	Program to translate C Web into C; default is `ctangle'.
RM	Command to remove a file; default is `rm -f'.

Here is a table of variables whose values are additional arguments for the programs above. The default values for all of these is the empty string, unless otherwise noted.

ARFLAGS	Flags to give the archive-maintaining program; default is `rv'.
ASFLAGS	Extra flags to give to the assembler when explicitly invoked on a `.s' or `.S' file.
CFLAGS	Extra flags to give to the C compiler.
CXXFLAGS	Extra flags to give to the C compiler.
COFLAGS	Extra flags to give to the RCS co program.
CPPFLAGS	Extra flags to give to the C preprocessor and programs, which use it *suchasCandFortrancompilers*.
FFLAGS	Extra flags to give to the Fortran compiler.
GFLAGS	Extra flags to give to the SCCS get program.

LDFLAGS	Extra flags to give to compilers when they are supposed to invoke the linker, `ld'.
LFLAGS	Extra flags to give to Lex.
YFLAGS	Extra flags to give to Yacc.
PFLAGS	Extra flags to give to the Pascal compiler.
RFLAGS	Extra flags to give to the Fortran compiler for Ratfor programs.
LINTFLAGS	Extra flags to give to lint.

NOTE− You can cancel all variables used by implicit rules with the '-R' or '--no-builtin-variables' option.

You can also define macros at the command line as shown below −

```
make CPP = /home/courses/cop4530/spring02
```