NAME

makefiles − users guide for compiling projects on different platforms

SYNOPSIS

make [target]
gmake [target]
smake [target]

Target may be one of:

.help	to get a list of possible targets and a short description.
all	to make the default targets
install	to make and install the default targets (see *INS_BASE* and *INS_KBASE* to learn how to modify the installation path).
ibins	to make and install a target in *SRCROOT/bins*
depend	to re-make dependency rules for all default targets. Note: All dependency files will automatically remade on the next run of make before they are included.
clean	to remove *core* files and all intermediate object files.
clobber	to remove the targets from *clean*, all dependency files and all final targets
distclean	to remove the targets from *clean* and *clobber* and all made files for all architectures.
tags	to make *vi*(1) compatible tags
TAGS	to make *emacs*(1) compatible tags
config	reserved for future use.
rmtarget	to remove the default target
relink	to remove the default target and remake it immediately. This can be used to change .e.g LD_RUN_PATH in the executable.

DESCRIPTION

Makefiles is a set of rules that allows compiling of structured projects with small and uniformly structured makefiles. All rules are located in a central directory. Compiling the projects on different platforms can be done without the need to modify any of the makefiles that are located in the projects directories.

Three make programs are supported: *Sunpro make*, *GNU make* and *smake*.

BSDmake could be supported if it supports pattern matching rules correctly.

The main design goal was to have a set of small and easy to read makefiles, each located in the project's leaf directory and therefore called *leaf*-makefile.

Each of these *leaf*-makefiles, in fact contains no rule at all. It simply defines some macros for the *make*-program and includes two files from a central make rule depository. The included files and the files that are recursively included define the rules that are needed to compile the project.

Each *leaf*-makefile is formed in a really simple way:

- It first defines two macros that define the relative location of the project's root directory and the name of the directory that contains the complete set of of rules and then includes the rule file *rules.top* from the directory that forms the central rule depository. You only have to edit the macro *SRCROOT* to reflect the relative location of the project's root directory.
- The next part of a *leaf*-makefile defines macros that describe the target and the source. You can only have one target per *leaf*-makefile. Of course, there may be many source files, that are needed to create that target. If you want to make more than one target in a specific directory, you have to put more than one makefile into that directory. This is the part of a makefile that describes a unique target. Edit this part to contain all source files, all local include files and all non global

compile time flags that are needed for your target. For a typical target this is as simple as filling in a form.

- Each *leaf*-makefile finally includes a file from the rules directory that contains rules for the appropriate type of target that is to be made from this *leaf*-makefile.

The makefile in each directory has to be called *Makefile*. If you want to have more than one makefile in a specific directory, you have to choose different names for the other makefiles.

There are rules for the following type of targets:

commands	The make rules for user level commands like *cat*, *ls* etc. are located in the file *rules.cmd*
drivers	The make rules for device drivers are located in the file *rules.drv*
libraries	The make rules for non shared libraries are located in the file *rules.lib*
shared libraries	The make rules for shared libraries are located in the file *rules.shl*
localized files	The make rules for localized files are located in the file *rules.loc*
nonlocalized files	The make rules for non localized files are located in the file *rules.aux*
shell scripts	The make rules for shell scripts (a variant of localized files) are located in the file *rules.scr*
manual pages	The make rules for manual pages (a variant of localized files) are located in the file *rules.man*
diverted makefiles	The make rules for projects that need to have more than one makefile in a specific directory are located in the file *rules.mks* It contains a rule that diverts to the listed sub makefiles. Each sub makefile may be of any type.
directories	The make rules for sub directories are located in the file *rules.dir*

Macros/Variables Used In Rules

The following is a description of the most important macros used within the make rules. *NOTE:* not all of them might be changed in a specific makefile. Carefully read the description and change only those macros that are intended to be used to change the behavior of the compilation.

ARCHDIR
: contains the location where object files and make targets will be placed. A typical name would be: *OBJ/sparc−sunos5−cc*
Do not change this macro.

ASFLAGS
: The flags that are used with the assembler.
Do not change this macro.

ASMFILES
: a list of assembler source files, to be specified in a leaf makefile.

ASOPTS
: The internal macro that contains the flags for the assembler. Change this macro if you want to change the behavior. Use: **ASOPTS= value** If you want to override the default value. If you want to override the default assembler flags from the command line call: **make 'ASOPTS=value'** Use: **ASOPTS += value** If you want to add flags to the default value from within a makefile.

ASOPTX
: may be used if you want to add flags to the assembler flags from the command line. Use: **make 'ASOPTX=value'**

C_ARCH
: this macro contains the c-compiler architecture name.
Do not change this macro.

CCOM
: This macro may be used from the command line to use a different c-compiler than the default. Use: **make 'CCOM=gcc'** to use *gcc* for the next run. Note: The value of *CCOM* must not necessarily be identical to the real name of the c-compiler. It refers to a filename

which contains the definition for that c-compiler. This hint applies as well to all other macros that deal with the name of the c-compiler. The only macro, that contains the real invocation name of the c-compiler, is *CC*. *CC* is set inside the machine dependent configuration file in the central rules depository and must not be changed elsewhere.

CFILES
: a list of C source files, to be specified in a leaf makefile.

CFLAGS
: The flags that are used with the c-compiler. Do not use flags that belong to the c-preprocessor in the *COPTOPT*, *COPTS* and *COPTX* macros.
Do not change this macro.

CLEAN_FILES
: A list of files (besides the object files) that will be removed with *make clean*.
Do not change this macro.

CLEAN_FILEX
: this macro may be used to define additional files that should be removed with *make clean*.

CLOBBER_FILEX
: this macro may be used to define additional files that should be removed with *make clobber*.

COPTOPT
: The c-compiler optimization flag. This flag is intended to be overridden either from a makefile or from command line.

COPTS
: The internal macro that contains the flags for the c-compiler. Change this macro if you want to change the behavior. Use: **COPTS= value** If you want to override the default value. If you want to override the default c-compiler flags from the command line call: **make 'COPTS=value'** Use: **COPTS += value** If you want to add flags to the default value from within a makefile.

COPTX
: may be used if you want to add flags to the c-compiler flags from the command line. Use: **make 'COPTX=value'**

CPPFLAGS
: The flags that are used with the c-preprocessor. This macro as well as: *CPPOPTS* and *CPPOPTX* are also used when compiling c-programs. They should contain only the following flags: *−Dname=value* , *−Uname* , *−Idirectory* and *−Ydirectory*.
Do not change this macro.

CPPOPTS
: The internal macro that contains the flags for the c-preprocessor. Change this macro if you want to change the behavior. Use: **CPPOPTS= value** If you want to override the default value. If you want to override the default c-preprocessor flags from the command line call: **make 'CPPOPTS=value'** Use: **CPPOPTS += value** If you want to add flags to the default value from within a makefile.

CPPOPTX
: may be used if you want to add flags to the c-preprocessor flags from the command line. Use: **make 'CPPOPTX=value'**

CURDIR
: This macro contains the name of the sub directory that is currently processed.
Do not change this macro.

CWARNOPTS
: This macro may be set in *$(DEFLTSROOT)/$(DEFLTSDIR)/Defaults* or *$(DEFLTSROOT)/$(DEFLTSDIR)/Defaults.$(O_ARCH)* to overwrite the default c-compiler warn options.

CWOPTS
: This macro is set inside the machine dependent configuration file in the central rules depository and must not be changed elsewhere. It contains the flags that set the default warning level for the c-compiler.

DEFINSGRP
: this macro may be set in the projects defaults file to set up the default group id for installation

DEFINSMODE
: this macro may be set in the projects defaults file to set up the default file permission for installation

DEFINSUSR
: this macro may be set in the projects defaults file to set up the default user id for installation

DEFUMASK
: this macro may be set in the projects defaults file to set up the default *umask* value for creating sub directories

DEFAULTSDIR
: this macro may be set from command line or from the shell environment if the user wants to use a different set of *Defaults* files that is not located in the directory **DEFAULTS** in the source root directory. This may be used to hold two or more set of defaults that differ e.g. in the installation path or the C-compiler.

DEFAULTSROOT
: this macro may be set in a leaf makefile, if that makefile wants to use a *Defaults* file that is not located in *$(DEFAULTSDIR)* in the source root. This may be used, if a sub tree in a big project needs a different setup.

DEFLTSDIR
: This is an internally used macro that is set from *$(DEFAULTSDIR)* or from the internal defaults.
Do not change this macro.

DEFLTSROOT
: This is an internally used macro that is set from *$(DEFAULTSROOT)* or from the internal defaults.
Do not change this macro.

DEFCCOM
: the default name of the c-compiler. This is usually set in the file *$(DEFLTSROOT)/$(DEFLTSDIR)/Defaults* or *$(DEFLTSROOT)/$(DEFLTSDIR)/Defaults.$(O_ARCH)*.

DEFINCDIRS
: a list of directories that are searched by default in addition to the system include directories. If this macro is not set, *$(SRCROOT)/include* is used.

DIRNAME
: this macro needs to be set in the makefile that is located in the *$(SRCROOT)* directory. The value should be either set to the name of the directory where the makefile is located or to **SRCROOT**. The value of this macro is updated automatically to follow the directory hierarchy. Do not change this macro in a make file other than the make file in the source root.

DIRS
: this macro needs to be set in a makefile that is located in a directory that contains diversion directories. It must contain a list of directory names where the diversions go to e.g. **DIRS= libfoo libbar libgarbage**.

HFILES
: for normal operation, *makefiles* will automatically find which include files are needed for compilation. However, if you want to create a tag file that includes tags for definitions that occur within include files, you should set *HFILES* to be a list of include files containing such definitions.

INCDIRS
: this macro will normally be a copy from *DEFINCDIRS*. You may however specify a different value in a leaf makefile or from command line. This will overwrite the defaults value.

INS_BASE
: this macro has to be specified in the file *$(DEFLTSROOT)/$(DEFLTSDIR)/Defaults* or *$(DEFLTSROOT)/$(DEFLTSDIR)/Defaults.$(O_ARCH)*. It must contain the path name of the root for installing general targets of the project. See *INSDIR*.

INS_KBASE
: this macro has to be specified in the file *$(DEFLTSROOT)/$(DEFLTSDIR)/Defaults* or *$(DEFLTSROOT)/$(DEFLTSDIR)/Defaults.$(O_ARCH)*. It must contain the path name of the root for installing kernel modules from the project. See *INSDIR*.

INSDIR this macro has to be specified in the leaf makefile. It must contain the path name of a directory relative to *INS_BASE* or *INS_KBASE*. The target will be installed into *$(INS_BASE)/$(INSDIR)* .

INSGRP this macro may be set in a leaf makefile to set up the group id for installation

INSMODE this macro may be set in a leaf makefile to set up the file permission for installation

INSUSR this macro may be set in a leaf makefile to set up the user id for installation

K_ARCH this macro contains the kernel/machine architecture for the target machine (e.g. *sun3 sun4c sun4m sun4u 9000−725 aviion*). It is set to the output of *uname −m* converted to lower case.
Do not change this macro.

KARCH this macro contains the kernel/machine architecture for the target machine (e.g. *sun3 sun4c sun4m sun4u 9000−725 aviion*). It is set to the output of *uname −m* converted to lower case.
Do not change this macro unless you want to do a cross compilation.

LDFLAGS The flags that are used with the linker.
Do not change this macro.

LDLIBS The internal macro that holds the libraries that are used while linking the target.
Do not change this macro.

LDOPTS The internal macro that contains the flags for the linker. Change this macro if you want to change the behavior. Use: **LDOPTS= value** If you want to override the default value. If you want to override the default linker flags from the command line call: **make 'LDOPTS=value'** Use: **LDOPTS += value** If you want to add flags to the default value from within a makefile.

LDOPTX may be used if you want to add flags to the linker flags from the command line. Use: **make 'LDOPTX=value'**

LDPATH the default library search path for the linker. This is usually set in the file *$(DEFLTSROOT)/$(DEFLTSDIR)/Defaults* or *$(DEFLTSROOT)/$(DEFLTSDIR)/Defaults.$(O_ARCH)*.

LIB_KVM a predefined macro that contains the operating system dependent name of a library that is needed by programs that read/write kernel virtual memory. Add *$(LIB_KVM)* to your list of libraries (e.g. **LIBS= −lintl $(LIB_KVM)**), if your target uses kvm.

LIB_MATH a predefined macro that contains the operating system dependent name of a library that is needed by programs that use routines of the math library. Add *$(LIB_MATH)* to your list of libraries (e.g. **LIBS= −lintl $(LIB_MATH)**), if your target uses math subroutines.

LIB_SOCKET a predefined macro that contains the operating system dependent name of a library that is needed by programs that use socket calls. Add *$(LIB_SOCKET)* to your list of libraries (e.g. **LIBS= −lintl $(LIB_SOCKET)**), if your target uses sockets. Note: this should (for portability reasons) even be done on systems that don't require a socket library because they have the socket interface inside libc.

LIBS_PATH this macro contains the path to a directory where those libraries are located, that have been build during a make run inside the current project.
Do not change this macro.

M_ARCH this macro contains the machine architecture for the target machine (e.g. *sun3 sun4 ip22 i86pc i586 9000−725 aviion*). It is set to the output of *arch* converted to lower case. On systems, where *M_ARCH* is not available, it is set to the content of *K_ARCH*.
Do not change this macro.

MK_FILES makefiles that divert into sub makes within the same directory must set *MK_FILES* to be a list of makefile names for the sub makes. Makefile names for sub makes should have a name that is build by adding *.mk* to the base name of the target that is defined inside the specific makefile.

MAKEPROG Set this macro to the name of your make program if it does not support the macro *MAKE_NAME*. The make program *smake* is able to identify itself. If you want to use a make program that is not able to identify itself and it's name is not *make*, set *MAKEPROG* to the name of the make program. Currently only *gmake* is supported as alternate value for *MAKEPROG*. If you want to use an unsupported make program you have to check if it supports the needed features for *makefiles*. You must write your own rule files for that make program. If you want to use *gmake*, you should do **setenv MAKEPROG gmake** before you start *gmake* or use a shell script that does this job for you.

MAKE This macro is set up by the *make* program. It contains a path name that is sufficient to recursively call the same *make* program again (either that last path component or the full path name of the make program).
Do not change this macro.

MAKEFLAGS
This macro is set up by the *make* program. It contains the command line flags, *make* is called with.
Do not change this macro.

MAKE_ARCH
This macro is currently set up by **smake** only. It contains the processor architecture of the target machine (e.g. mc68020, sparc, pentium).
Do not change this macro.

MAKE_BRAND
This macro is currently set up by **smake** only. It contains the brand of the target machine (e.g. Sun_Microsystems(e.g.).
Do not change this macro.

MAKE_HOST
This macro is currently set up by **smake** only. It contains the host name of the target machine (e.g. duffy, sherwood, ghost).
Do not change this macro.

MAKE_MACH
This macro is currently set up by **smake** only. It contains the kernel architecture of the target machine (e.g. sun3, sun4c, sun4m, sun4u).
Do not change this macro.

MAKE_MODEL
This macro is currently set up by **smake** only. It contains the model name of the target machine (e.g. SUNW,SPARCstation-20).
Do not change this macro.

MAKE_M_ARCH
This macro is currently set up by **smake** only. It contains the machine architecture of the target machine (e.g. sun3, sun4).
Do not change this macro.

MAKE_NAME
This macro is currently set up by **smake** only. It contains the official name of the make program (e.g. make, smake, gmake).
Do not change this macro.

MAKE_OS This macro is currently set up by **smake** only. It contains the operating system name of the target machine (e.g. sunos, linux, dgux).
Do not change this macro.

MAKE_OSDEFS
: This macro is currently set up by **smake** only. It contains operating system specific defines for the compiler (e.g. -D__SVR4).
Do not change this macro.

MAKE_OSREL
: This macro is currently set up by **smake** only. It contains the operating system release name of the target machine (e.g. 5.5, 4.1.1).
Do not change this macro.

MAKE_OSVERSION
: This macro is currently set up by **smake** only. It contains the operating system version of the target machine (e.g. generic).
Do not change this macro.

MAKE_SERIAL
: This macro is currently set up by **smake** only. It contains the serial number of the target machine (e.g. 1920098175).
Do not change this macro.

MANDIR
: all makefiles for manual pages must set this macro to the path name relative to *INS_BASE* where the manual page root dir for the project should be. Possible values for english manual pages are *man* or *share/man*, possible values for german manual pages are *man/de* or *share/man/de*.

MANFILE
: makefiles for manual pages set this macro to the name of the troff source file for the manual page

MANSECT
: makefiles for manual pages set this macro to the macro name that contains the real section name for this manual page. This is necessary because bsd based system have different naming conventions than system V based systems. See below for a valid list of manual section macros.

MANSECT_ADMIN
: This macro contains the name of the sub directory for administrative commands and maintenance procedures.
Do not change this macro.

MANSECT_CMD
: This macro contains the name of the sub directory for general user commands.
Do not change this macro.

MANSECT_DEMOS
: This macro contains the name of the sub directory for demo commands.
Do not change this macro.

MANSECT_DEVICE
: This macro contains the name of the sub directory for user level device interfaces.
Do not change this macro.

MANSECT_DRIVER
: This macro contains the name of the sub directory for kernel level device driver interfaces.
Do not change this macro.

MANSECT_FILEFORM
: This macro contains the name of the sub directory for file formats.
Do not change this macro.

MANSECT_GAMES
: This macro contains the name of the sub directory for games.
Do not change this macro.

MANSECT_HDR
: This macro contains the name of the sub directory for header files.
Do not change this macro.

MANSECT_LIB
: This macro contains the name of the sub directory for library function interfaces.
Do not change this macro.

MANSECT_MACROS
: This macro contains the name of the sub directory for troff macros.
Do not change this macro.

MANSECT_NETWORK
: This macro contains the name of the sub directory for user level network interfaces.
Do not change this macro.

MANSECT_SYSCALL
: This macro contains the name of the sub directory for system call interfaces. Do not change this macro.

MANSECT_TABLES
: This macro contains the name of the sub directory for tables. Do not change this macro.

MANSTYLE
: This macro contains the name that is used to find the right ordering conventions for manual pages. Do not change this macro.

MANSUFFIX
: makefiles for manual pages set this macro to the macro name that contains the real suffix for this manual page. This is necessary because bsd based system have different naming conventions than system V based systems. See below for a valid list of manual suffix macros.

MANSUFF_ADMIN
: This macro contains the name of the file suffix for administrative commands and maintenance procedures.
Do not change this macro.

MANSUFF_CMD
: This macro contains the name of the file suffix for general user commands.
Do not change this macro.

MANSUFF_DEMOS
: This macro contains the name of the file suffix for demo commands.
Do not change this macro.

MANSUFF_DEVICE
: This macro contains the name of the file suffix for user level device interfaces.
Do not change this macro.

MANSUFF_DRIVER
: This macro contains the name of the file suffix for kernel level device driver interfaces.
Do not change this macro.

MANSUFF_FILEFORM
: This macro contains the name of the file suffix for file formats.
Do not change this macro.

MANSUFF_GAMES
: This macro contains the name of the file suffix for games.
Do not change this macro.

MANSUFF_HDR
: This macro contains the name of the file suffix for header files.
Do not change this macro.

MANSUFF_LIB
: This macro contains the name of the file suffix for library function interfaces.
Do not change this macro.

MANSUFF_MACROS
: This macro contains the name of the file suffix for troff macros.
Do not change this macro.

MANSUFF_NETWORK
: This macro contains the name of the file suffix for user level network interfaces.
Do not change this macro.

MANSUFF_SYSCALL
: This macro contains the name of the file suffix for system call interfaces. Do not change this macro.

MANSUFF_TABLES
: This macro contains the name of the file suffix for tables. Do not change this macro.

MARCH
: this macro contains the machine architecture for the target machine (e.g. *sun3 sun4 ip22 i86pc i586 9000−725 aviion*). It is set to the output of *arch* converted to lower case. On systems, where *M_ARCH* is not available, it is set to the content of *K_ARCH*.
Do not change this macro unless you want to do a cross compilation.

O_ARCH
: this macro contains the name of the operating system converted to lower case. It is usually the output of: *uname −s*. It may contain a modified name if the compilation rules for different version of the operating system differ (e.g. on *SunOS* and *Solaris*, the official operation system name in both cases is *SunOS*).
Do not change this macro.

OARCH
: this macro contains the object architecture that is used for architecture dependent sub directories. It is set to: *$(PARCH)−$(O_ARCH)−$(C_ARCH)*.
Do not change this macro.

OARCHDIR
: this macro contains the concatenation of *OBJ/* and *$(OARCH)*.
Do not change this macro.

OFILES
: this macro contains the list of objects that are the the dependency list for the target. It is constructed from the list of assembler source files, c source files, lex source files and yacc source files.
Do not change this macro.

OINCSDIR
: this macro contains the concatenation of *$(SRCROOT)/incs/* and *$(OARCH)*. It is the location where include files that are made within a make run and subject to global use will be placed.
Do not change this macro.

OLIBSDIR
: this macro contains the concatenation of *$(SRCROOT)/libs/* and *$(OARCH)*. It is the location where libraries that are made within a make run will be placed.
Do not change this macro.

OSDEFS
: this macro contains operating system specific c-preprocessor definitions.
Do not change this macro.

OSNAME
: the unmodified name of the operating system converted to lower case. See *O_ARCH*.
Do not change this macro.

OSREL
: the release of the operating system.
Do not change this macro.

P_ARCH
: this macro contains the processor architecture for the target machine (e.g. *mc68020 mc88200 sparc pentium*). It is set to the output of *uname −p* converted to lower case. On systems, where *P_ARCH* is not available, it is set to the content of *K_ARCH*.

Do not change this macro.

PARCH
: this macro contains the processor architecture for the target machine (e.g. *mc68020 mc88200 sparc pentium*). It is set to the output of *uname −p* converted to lower case. On systems, where *P_ARCH* is not available, it is set to the content of *K_ARCH*.
Do not change this macro unless you want to do a cross compilation.

RULESDIR
: the value of this macro must be set before the rules file *$(SRCROOT)/$(RULESDIR/rules.top* is included. If you want to test the behavior of a modified version of the *makefiles* in *RULESDIR*, put a copy into an alternate directory, modify one or more files in it and then use make with a different value of *RULESDIR*, that points to the alternate directory.

RUNPATH
: is evaluated on systems, where *LD_RUN_PATH* is supported. It contains the default library search path for dynamic linked targets on runtime. This search path will be stored inside the target. This is usually set in the file *$(DEFLTSROOT)/$(DEFLTSDIR)/Defaults* or *$(DEFLTSROOT)/$(DEFLTSDIR)/Defaults.$(O_ARCH)*. Note that older systems will use *LD_LIBRARY_PATH* for this purpose too.

SCRFILE
: this macro must be set in a leaf makefile for shell scripts to define the source for that script file.

SRCFILE
: this macro must be set in a leaf makefile for localized files to define the source for that file.

SRCROOT
: this macro contains the relative position to the project's source root directory. The value of this macro must be set before the rules file *$(SRCROOT)/$(RULESDIR/rules.top* is included. *SRCROOT* should be set to *../..* if the appropriate leaf makefile is located two directory levels below the source route.

SUBARCHDIR
: may be set to put the object files and the target into a different directory than usual. *SUBARCHDIR* will modify the value of *ARCHDIR*. If you want to make a target that is compiled for *dbx* you may use: *make COPTX=−g SUBARCHDIR=−dbx*.

TARGET
: all makefiles, that are not referring to library targets or sub makes / sub dir makes, must define *TARGET* to be the output file name of the final link operation.

TARGETLIB
: all makefiles that are used to make libraries must define *TARGETLIB* to be the central part of the target library. If you want to make e.g. *libfoo* define *TARGETLIB= foo*. This is needed to allow operating systems to have different naming conventions for libraries. If you are making a non shared library, the example above would result in a filename: *libfoo.a* for the real target.

TARGETMAN
: this macro must be set in a leaf makefile for manual pages to define the base name for that manual page (not including the suffix).

XMK_FILE
: makefiles that want to install manual pages should set *XMK_FILE* to **Makefile.man** to avoid having to install a diversion make file. The make file found in *XMK_FILE* will be processed only if *make* is called with the target *install*.

GETTING STARTED

To set up a new project, first copy the *RULES* and *TEMPLATES* directories and all its content into the project's root directory.

Then copy a top level makefile, a *Defaults* file and a *Targetdirs.archname* file into the project's root directory.

Then create the following directories: *cmd*, *lib*, *include*.

Now start creating target directories e.g below *cmd* and *lib*. Don't forget to create a makefile and an appropriate *Targetdirs.archname* file on each node directory.

EXAMPLES

If you want to set up a private copy of parts of a project, you should choose a directory inside your directory tree that will become the shadow projects source root directory. Then create symbolic links named *RULES* and *TEMPLATES* to the real source root. If you don't want to modify the global include files, create a symbolic link to the *include* directory too, else copy the include directory and its content. copy the top level makefile, the *Defaults* file and the *Targetdirs.archname* file.

Finally copy the desired sub tree together with all makefiles and the *Targetdirs.archname* files that are located in the directories above your part of the project that you want to have separately.

FILES

.../RULES/*
.../DEFAULTS/*
.../TARGETS/*
.../TEMPLATES/*

SEE ALSO

makerules(4), **make**(1), **gmake**(1), **smake**(1).

If you want to know, how to add new rules or how to add support for new operating systems or compilers look into **makerules**(4).

DIAGNOSTICS

Diagnostic messages depend on the make program. Have a look at the appropriate man page.

NOTES

The scope of this manual is only the usage of **leaf makefiles**. If you want to make changes to the **make rules** have a look at **makerules**(4).

makefiles can be used with *Sunpro make*, *Gnu make* and *smake*. Although Gnu make runs on many platforms, it has no useful debug output.

Use *Sunpro make* or *smake* if you have problems with a makefile. *Sunpro make* and *smake*, both have a −D flag, that allows you to watch the makefiles after the first expansion. Use this option, if you are in doubt if your makefile gets expanded the right way and if the right rules are included. There is also a −d option that gives debugging output while make is running. If you want more output, use −dd, −ddd and so on.

Smake has an option −xM that shows you the include dependency for make rules.

The following is a list of all macros that are used in *makefiles*. Do not use them unless their meaning has been explained before.

−O_ARCH, **.OBJDIR**, **.SEARCHLIST**, **ALLTARGETS**, **AR**, **ARCH**, **ARCHDIR**, **ARCHDIRX**, **ARCH_DONE**, **ARFLAGS**, **AS**, **ASFLAGS**, **ASMFILES**, **ASOPTS**, **ASOPTX**, **CC**, **CCOM**, **CCOM_DEF**, **CFILES**, **CFLAGS**, **CHGRP**, **CHMOD**, **CHOWN**, **CLEAN_FILES**, **CLEAN_FILEX**, **CLOBBER_FILEX**, **COPTDYN**, **COPTGPROF**, **COPTOPT**, **COPTS**, **COPTX**, **CPP**, **CPPFLAGS**, **CPPOPTS**, **CPPOPTX**, **CTAGS**, **CURDIR**, **CWARNOPTS**, **CWOPTS**, **C_ARCH**, **DEFAULTSDIR**, **DEFAULTSROOT**, **DEFCCOM**, **DEFCCOM_DEF**, **DEFINCDIRS**, **DEFINCDIRS_DEF**, **DEFINSGRP**, **DEFINSMODE**, **DEFINSUSR**, **DEFUMASK**, **DEF_ROOT**, **DEP_DEP**, **DEP_FILE**, **DEP_SUFFIX**, **DIRNAME**, **DIRS**, **DYNLD**, **ETAGS**, **FLOAT_OPTIONS**, **HFILES**, **HOSTNAME**, **INCDIRS**, **INSDIR**, **INSGRP**, **INSGRP_DEF**, **INSMODE**, **INSMODE_DEF**, **INSTALL**, **INSUSR**, **INSUSR_DEF**, **INS_BASE**, **INS_KBASE**, **KARCH**, **KARCH_DEF**, **KDEFINES**, **K_ARCH**, **LD**, **LDCC**, **LDFLAGS**, **LDLIBS**, **LDOPTDYN**, **LDOPTS**, **LDOPTX**, **LDPATH**, **LIBS**, **LIBS_PATH**, **LIBX**, **LIB_KVM**, **LIB_MATH**, **LIB_PREFIX**, **LIB_SOCKET**, **LIB_SUFFIX**, **LN**, **LNDYNLIB**, **LOCALIZE**, **LORDER**, **MAKE**, **MK_FILES**, **MAKEPROG**, **MAKE_ARCH**, **MAKE_HOST**, **MAKE_MACH**, **MAKE_M_ARCH**, **MAKE_NAME**, **MAKE_OS**, **MAKE_OSDEFS**, **MAKE_OSREL**, **MANDIR**, **MANFILE**, **MANSECT**, **MANSECT_ADMIN**, **MANSECT_CMD**, **MANSECT_DEMOS**, **MANSECT_DEVICE**, **MANSECT_DRIVER**, **MANSECT_FILEFORM**, **MANSECT_GAMES**, **MANSECT_HDR**, **MANSECT_LIB**, **MANSECT_MACROS**, **MANSECT_NETWORK**, **MANSECT_SYSCALL**, **MANSECT_TABLES**, **MANSTYLE**, **MANSUFFIX**, **MANSUFF_ADMIN**, **MANSUFF_CMD**, **MANSUFF_DEMOS**, **MANSUFF_DEVICE**, **MANSUFF_DRIVER**, **MANSUFF_FILEFORM**, **MANSUFF_GAMES**, **MANSUFF_HDR**, **MANSUFF_LIB**, **MANSUFF_MACROS**, **MANSUFF_NETWORK**, **MANSUFF_SYSCALL**, **MANSUFF_TABLES**, **MARCH**, **MARCH_DEF**, **MKDEP**, **MKDEP_OUT**, **MKDIR**, **MV**, **M_ARCH**, **OARCH**, **OARCHDIR**, **OFILES**, **OINCSDIR**, **OLIBSDIR**, **OSDEFS**, **OSNAME**, **OSREL**, **O_ARCH**, **PALLDEP_FILE**, **PARCH**, **PARCH_DEF**, **PASMFILES**, **PDEP_FILE**, **PLOFILES**, **POFILES**, **PTARGET**, **PTARGET_BASE**, **PYOFILES**, **P_ARCH**, **RANLIB**, **RM**, **RMDEP**, **RMTARGET**, **RM_FORCE**, **RM_RECURS**, **RM_RF**, **RULESDIR**, **RUNPATH**, **SCRFILE**, **SHELL**, **SHL_SUFFIX**, **SRCFILE**, **SRCLIBS**, **SRCROOT**, **SUBARCH**, **SUBARCHDIR**, **SYMLINK**, **TAGS**, **TARGET**, **TARGETLIB**, **TARGETMAN**, **TARGET_BASE**, **TARGET_PATH**, **TSORT**, **UMASK**, **UMASK_DEF**, **UMASK_VAL**, **XARCH**, **XK_ARCH**, **XMK_FILE**, **XMAKEPROG**, **XM_ARCH**, **XP_ARCH**, **_CCOM**, **_DEFAULTSDIR**, **_DEFCCOM**, **_DEFINSGRP**, **_DEFINSMODE**, **_DEFINSUSR**, **_DEFUMASK**, **_DIRNAME**, **_INCDIRS**, **_MAKEPROG**, **_MARCH**, **_M_ARCH**, **_O_ARCH**, **_PARCH**, **_P_ARCH**, **_UNIQ**, **__CCOM**, **__DEFAULTSDIR**, **__DEFCCOM**, **__DEFINSGRP**, **__DEFINSMODE**, **__DEFINSUSR**, **__DEFUMASK**, **__DIRNAME**, **__INCDIRS**, **__MAKEPROG**, **__MARCH**, **__M_ARCH**, **__PARCH**, **__P_ARCH**,

BUGS

Source Tree Hierarchy

The following outline gives a quick tour through a typical source hierarchy:

.../ root directory of the source tree

Makefile
the top Makefile

Targetdirs
a file containing a list of directories that are needed for that project. If the system needs different target lists depending on the target system architecture , use target specific files in **.../TARGETS/**

. . .

.../RULES/
the location of makefiles (included rules)

rules.top
the mandatory include rules (needed to setup basic rules)

rules.aux
rules needed to install a non localized auxiliary file

rules.cmd
rules needed to make an ordinary command (like /bin/sh)

rules.drv
rules needed to make a device driver

rules.lib
rules needed to make a standard (nonshared) library

rules.loc
rules needed to install a localized auxiliary file

rules.man
rules needed to install a localized manual page

rules.scr
rules needed to install a localized shell script

rules.shl
rules needed to make a shared library

rules.mks
rules needed to make more than one target in a specific directory

rules.dir
rules needed to make targets that are located in sub directories to the current directory

. . .

.../DEFAULTS/
default definitions for various target architectures are located in this directory. Templates for some architectures can be found in the *.../TEMPLATES/* directory.

Defaults
default definitions for that source tree. System dependent definitions are in **.../DEFAULTS/Defaults.***

.../TARGETS/
target list definitions for various target architectures are located in this directory.

.../TEMPLATES/
templates that should be used inside the project (rename to Makefile, if it is the only makefile in that directory, rename to *target.mk,* if there is more than one target in that directory)

Defaults
Defaults file for the source root directory

Defaults.linux
Defaults file for *linux*. This should be installed in the **.../DEFAULTS/** directory.

Makefile.root
Makefile for the source root directory

Makefile.aux
Makefile for a non localized auxiliary file

Makefile.cmd
Makefile for an ordinary command (like /bin/sh)

Makefile.lib
Makefile for a standard (nonshared) library

Makefile.loc
Makefile for a localized auxiliary file

Makefile.man
Makefile for a localized manual page

Makefile_de.man
Makefile for a localized manual page in the german locale

Makefile.scr
Makefile for a localized shell script
Makefile.shl
Makefile for a shared library
Makefile.drv
Makefile for a device driver
Makefile.mks
Makefile for more than one target in a specific directory
Makefile.dir
Makefile for targets that are located in sub directories to the current directory
. . .

.../cmd/
source tree for normal commands
Makefile
the makefile for the *cmd* sub directory
Targetdirs.sun4m
a file containing a list of directories like *myprog* (see below) that are needed for that specific architecture.
myprog/
directory where the sources for a specific command are located
Makefile
makefile for *myprog*
Makefile.man
makefile for the manual page of *myprog*
mprog.c
source for myprog
mprog.tr
troff source for the manual page of myprog
OBJ/ directory where system specific sub directories are located
sparc−sunos5−cc/
directory for binaries that belong to a specific system
. . .
. . .
. . .

.../lib/ directory where the sources for a libraries are located
Makefile
the makefile for the *lib* sub directory
Targetdirs.sun4m
a file containing a list of directories like *libfoo* (see below) that are needed for that specific architecture.
libfoo/ directory where all source files for libfoo are located
. . .

.../kernel
directory for kernel modules
Makefile
the makefile for the *kernel* sub directory
Targetdirs.sun4m
a file containing a list of directories like *drv* (see below) that are needed for that specific architecture.
drv/ directory where drivers are located
Makefile
the makefile for the *drv* sub directory

Targetdirs.sun4m
a file containing a list of directories like *mydrv* (see below) that are needed for that specific architecture.

mydrv/
source for a specific driver

. . .

. . .

.../include
directory for global include files that are used in that project

.../bins directory for binary programs that are created/needed while compiling the project

sparc−sunos5−cc/
directory for binaries that belong to a specific system

. . .

.../libs directory for libraries that are created/needed while compiling the project

sparc−sunos5−cc/
directory for libraries that belong to a specific system

. . .

.../incs directory for include files that are created/needed while compiling the project

sparc−sunos5−cc/
directory for include files that belong to a specific system

. . .

. . .

AUTHOR

Jörg Schilling
Seestr. 110
D-13353 Berlin
Germany

Mail bugs and suggestions to:

joerg@schily.isdn.cs.tu-berlin.de or **js@cs.tu-berlin.de** or **jes@fokus.gmd.de**