NAME

crunchgen - generates build environment for a crunched binary

SYNOPSIS

crunchgen [**-foql**] [**-h** *makefile-header-name*] [**-m** *makefile-name*] [**-p** *obj-prefix*] [**-c** *c-file-name*] [**-e** *exec-file-name*] *conf-file*

DESCRIPTION

A crunched binary is a program made up of many other programs linked together into a single executable. The crunched binary **main**() function determines which component program to run by the contents of *argv[0]*. The main reason to crunch programs together is for fitting as many programs as possible onto an installation or system recovery floppy.

The **crunchgen** utility reads in the specifications in *conf-file* for a crunched binary, and generates a *Makefile* and accompanying top-level C source file that when built creates the crunched executable file from the component programs. For each component program, **crunchgen** can optionally attempt to determine the object (.o) files that make up the program from its source directory *Makefile*. This information is cached between runs. The **crunchgen** utility uses the companion program crunchide(1) to eliminate link-time conflicts between the component programs by hiding all unnecessary symbols.

The **crunchgen** utility places specific requirements on package *Makefile*s which make it unsuitable for use with non-BSD sources. In particular, the *Makefile* must contain the target **depend**, and it must define all object files in the variable *OBJS*. In some cases, you can use a fake *Makefile*: before looking for *Makefile* in the source directory *foo*, **crunchgen** looks for the file *Makefile.foo* in the current directory.

After **crunchgen** is run, the crunched binary can be built by running "make -f <conf-name>.mk". The component programs' object files must already be built. An **objs** target, included in the output makefile, will run make(1) in each component program's source dir to build the object files for the user. This is not done automatically since in release engineering circumstances it is generally not desirable to be modifying objects in other directories.

The options are as follows:

-c *c-file-name*
: Set output C file name to *c-file-name*. The default name is <*conf-name*>.*c*.

-e *exec-file-name*
: Set crunched binary executable file name to *exec-file-name*. The default name is <*conf-name*>.

-f Flush cache. Forces the recalculation of cached parameters.

-l List names. Lists the names this binary will respond to.

-h *makefile-header-name*

Set the name of a file to be included at the beginning of the *Makefile*s generated by **crunchgen**. This is useful to define some make variables which might affect the behavior of make(1) and are annoying to pass through environment variables.

-m *makefile-name*

Set output *Makefile* name to *makefile-name*. The default name is *<conf-name>.mk*.

-o Add "make obj" rules to each program make target.

-p *obj-prefix*

Set the pathname to be prepended to the **srcdir** when computing the **objdir**. If this option is not present, then the prefix used is the content of the MAKEOBJDIRPREFIX environment variable, or */usr/obj*.

-q Quiet operation. Status messages are suppressed.

CRUNCHGEN CONFIGURATION FILE COMMANDS

The **crunchgen** utility reads specifications from the *conf-file* that describe the components of the crunched binary. In its simplest use, the component program names are merely listed along with the top-level source directories in which their sources can be found. The **crunchgen** utility then calculates (via the source makefiles) and caches the list of object files and their locations. For more specialized situations, the user can specify by hand all the parameters that **crunchgen** needs.

The *conf-file* commands are as follows:

srcdirs *dirname ...*

A list of source trees in which the source directories of the component programs can be found. These dirs are searched using the BSD "*<source-dir>/<progname>/*" convention. Multiple **srcdirs** lines can be specified. The directories are searched in the order they are given.

progs *progname ...*

A list of programs that make up the crunched binary. Multiple **progs** lines can be specified.

libs *libspec ...*

A list of library specifications to be included in the crunched binary link. Multiple **libs** lines can

be specified.

libs_so *libspec* ...
A list of library specifications to be dynamically linked in the crunched binary. These libraries will need to be made available via the run-time link-editor rtld(1) when the component program that requires them is executed from the crunched binary. Multiple **libs_so** lines can be specified. The **libs_so** directive overrides a library specified gratuitously on a **libs** line.

buildopts *buildopts* ...
A list of build options to be added to every make target.

ln *progname linkname*
Causes the crunched binary to invoke *progname* whenever *linkname* appears in *argv[0]*. This allows programs that change their behavior when run under different names to operate correctly.

To handle specialized situations, such as when the source is not available or not built via a conventional *Makefile*, the following **special** commands can be used to set **crunchgen** parameters for a component program.

special *progname* **srcdir** *pathname*
Set the source directory for *progname*. This is normally calculated by searching the specified **srcdirs** for a directory named *progname*.

special *progname* **objdir** *pathname*
Set the *obj* directory for *progname*. The *obj* directory is normally calculated by looking for a directory whose name is that of the source directory prepended by one of the following components, in order of priority: the **-p** argument passed to the command line; or, the value of the MAKEOBJDIRPREFIX environment variable, or */usr/obj*. If the directory is not found, the **srcdir** itself becomes the **objdir**.

special *progname* **buildopts** *buildopts*
Define a set of build options that should be added to make(1) targets in addition to those specified using **buildopts** when processing *progname*.

special *progname* **objs** *object-file-name* ...
Set the list of object files for program *progname*. This is normally calculated by constructing a temporary makefile that includes "**srcdir**/*Makefile*" and outputs the value of *$(OBJS)*.

special *progname* **objpaths** *full-pathname-to-object-file* ...
Sets the pathnames of the object files for program *progname*. This is normally calculated by

prepending the **objdir** pathname to each file in the **objs** list.

special *progname* **objvar** *variable_name*
Sets the name of the make(1) variable which holds the list of object files for program *progname*. This is normally *OBJS* but some *Makefile*s might like to use other conventions or prepend the program's name to the variable, e.g., *SSHD_OBJS*.

special *progname* **lib** *library-name* ...
Specifies libraries to be linked with object files to produce *progname.lo*. This can be useful with libraries which redefine routines in the standard libraries, or poorly written libraries which reference symbols in the object files.

special *progname* **keep** *symbol-name* ...
Add specified list of symbols to the keep list for program *progname*. An underscore ('_') is prepended to each symbol and it becomes the argument to a **-k** option for the crunchide(1) phase. This option is to be used as a last resort as its use can cause a symbol conflict, however in certain instances it may be the only way to have a symbol resolve.

special *progname* **ident** *identifier*
Set the *Makefile*/C identifier for *progname*. This is normally generated from a *progname*, mapping '-' to '_' and ignoring all other non-identifier characters. This leads to programs named "foo.bar" and "foobar" to map to the same identifier.

Only the **objpaths** parameter is actually needed by **crunchgen**, but it is calculated from **objdir** and **objs**, which are in turn calculated from **srcdir**, so is sometimes convenient to specify the earlier parameters and let **crunchgen** calculate forward from there if it can.

The makefile produced by **crunchgen** contains an optional **objs** target that will build the object files for each component program by running make(1) inside that program's source directory. For this to work the **srcdir** and **objs** parameters must also be valid. If they are not valid for a particular program, that program is skipped in the **objs** target.

EXAMPLES

Here is an example **crunchgen** input conf file, named "*kcopy.conf*":

```
srcdirs /usr/src/bin /usr/src/sbin

progs test cp echo sh fsck halt init mount umount myinstall
progs anotherprog
ln test [       # test can be invoked via [
```

```
ln sh -sh       # init invokes the shell with "-sh" in argv[0]

special myprog objpaths /homes/leroy/src/myinstall.o # no sources

special anotherprog -DNO_FOO WITHOUT_BAR=YES

libs -lutil -lcrypt
```

This conf file specifies a small crunched binary consisting of some basic system utilities plus a homegrown install program "*myinstall*", for which no source directory is specified, but its object file is specified directly with the **special** line.

Additionally when "*anotherprog*" is built the arguments

```
-DNO_FOO WITHOUT_BAR=YES
```

are added to all build targets.

The crunched binary "*kcopy*" can be built as follows:

```
% crunchgen -m Makefile kcopy.conf    # gen Makefile and kcopy.c
% make objs           # build the component programs' .o files
% make               # build the crunched binary kcopy
% kcopy sh           # test that this invokes a sh shell
$                             # it works!
```

At this point the binary "*kcopy*" can be copied onto an install floppy and hard-linked to the names of the component programs.

Note that if the **libs_so** command had been used, copies of the libraries so named would also need to be copied to the install floppy.

SEE ALSO

crunchide(1), make(1), rtld(1)

AUTHORS

The **crunchgen** utility was written by James da Silva <*jds@cs.umd.edu*>.

The **libs_so** keyword was added in 2005 by Adrian Steinmann <*ast@marabu.ch*> and Ceri Davies <*ceri@FreeBSD.org*>.

CAVEATS

While **crunchgen** takes care to eliminate link conflicts between the component programs of a crunched binary, conflicts are still possible between the libraries that are linked in. Some shuffling in the order of libraries may be required, and in some rare cases two libraries may have an unresolvable conflict and thus cannot be crunched together.

Some versions of the BSD build environment do not by default build the intermediate object file for single-source file programs. The "make objs" must then be used to get those object files built, or some other arrangements made.