

**NAME**

regen – regenerate system directories

**DESCRIPTION**

There are several system directories that contain various commands and archive libraries. These include */bin*, */usr/bin*, */lib*, */usr/lib*, */etc*, */usr/hasp*, and */usr/fort*. To facilitate the regeneration of these directories, the *make(I)* command is used. Each directory contains the *make* description files used to build, update, or regenerate the directory. If one description file suffices, it is named *.makefile*; otherwise the files are named *.makefile1*, *.makefile2*, ...

For best results, you should be logged in as “root” (or as super-user).

For example, the following will recompile any */bin* commands that are out-of-date with respect to their corresponding source files:

```
chdir /bin
make -f .makefile1
make -f .makefile2
make -f .makefile3
```

As another example, the following forces the recompilation of any */bin* commands that call *ctime(III)*, in addition to recompiling anything that is out-of-date in */bin*:

```
chdir /bin
make -f .makefile1 CTIME=RC
make -f .makefile2 CTIME=RC
make -f .makefile3 CTIME=RC
```

The section on “Time Zones” below explains why this is useful. Similar arguments force recompilation of anything that uses the C compiler (CCDEP=RC), the assembler (ASDEP=RC), the *yacc(I)* processor (YACCDDEP=RC), the “-ls” library (LSDEP=RC), the “-lpw” library (LPWDEP=RC), the “-lpw” library (LpwDEP=RC), and the UNIX operating system “include” files (SYSDEP=RC).

**Time Zones:**

Commands such as *ls(I)* and *date(I)* give Eastern times (EST or EDT). This has been known to annoy non-East Coast (USA) and non-USA users. To get your local time, you must change *ctime(III)*, which converts the GMT *time(II)* time-stamps to local time, and then regenerate all commands that use *ctime*. To be precise, you must:

— Change *ctime(III)*. The source currently lives in */sys/source/s4/ctime.c*, and the object resides in the standard C library, */lib/libc.a*.

— Regenerate the standard C library, as follows:

```
chdir /lib
make -f .makefile1
make -f .makefile2
make -f .makefile3
make -f .makefile4
```

This works because, as a result of the change to the source for *ctime(III)*, */lib/libc.a* is now out-of-date with respect to its source, namely the *ctime* part.

— Regenerate all directories, using the CTIME=RC option on the *make* command line:

```

chdir /bin
make -f .makefile1 CTIME=RC
make -f .makefile2 CTIME=RC
make -f .makefile3 CTIME=RC
chdir /usr/bin
make -f .makefile1 CTIME=RC
make -f .makefile2 CTIME=RC
make -f .makefile3 CTIME=RC
make -f .makefile4 CTIME=RC
chdir /lib
make -f .makefile1 CTIME=RC
make -f .makefile2 CTIME=RC
make -f .makefile3 CTIME=RC
make -f .makefile4 CTIME=RC
chdir /usr/lib
make -f .makefile1 CTIME=RC
make -f .makefile2 CTIME=RC
chdir /etc
make -f .makefile CTIME=RC
chdir /usr/hasp
make -f .makefile CTIME=RC
chdir /usr/fort
make -f .makefile CTIME=RC

```

The full directory list (as of this moment!) is given above for completeness. Some directories might not have anything that uses *ctime*.

Of course, you should test your new *ctime* before installing it in the standard C library. See also the "BUGS" section below.

#### **/bin:**

Regeneration of */bin* is straightforward, except for *as*(I), *ld*(I), and *cc*(I). The *make* description files for */bin* do *not* automatically regenerate *cc*; instead, a separate procedure is used (see "cc" below).

*As* and *ld* are necessary to regenerate themselves. That is, one cannot re-assemble *as* unless a working *as* already exists. Thus, if something goes wrong and a bad *as* or *ld* is generated, you'll have to retrieve a working version from a backup tape.

The moral: before regenerating */bin*, be sure that you either have a good backup tape, or else have emergency copies of *as* and *ld* somewhere.

#### **/usr/bin:**

*.makefile1* and *.makefile2* regenerate the basic */usr/bin* commands; *.makefile3* regenerates the SCCS commands; *.makefile4* regenerates *eqn*(I) and *troff*(I). If you do not have the PWB/UNIX *troff*(I) and *eqn*(I) package, *.makefile4* will fail.

#### **/lib:**

As with *as* and *ld* in */bin*, *as2* in */lib* ("pass 2" of *as*) can't be regenerated if it doesn't already exist. Also, *c0*, *c1*, *c2*, and *cpp* in */lib* are parts of the C compiler, and the *make* description files for */lib* do not automatically regenerate them (see "cc" below).

#### **/usr/lib:**

*.makefile1* regenerates the basic */usr/lib* modules; *.makefile2* regenerates tables used by *eqn*(I) and *troff*(I). If you do not have the PWB/UNIX *troff*(I) and *eqn*(I) package, *.makefile2* will fail.

**cc:**

*/bin/cc* is just the tip of the C compiler iceberg. The real work is done by *c0*, *c1*, *c2*, and *cpp* in */lib*. Since these programs work together, they must be updated carefully. Furthermore, since the C compiler is written in C, if you install a bad C compiler you'll have to retrieve a working version from a backup tape.

Therefore, the *make* procedures for */bin* and */lib* do not attempt to regenerate the C compiler. Instead, you must do it explicitly, by:

```
chdir /sys/c/c
make -f makefile install clean
```

That creates *cc*, *c0*, *c1*, *c2*, and *cpp* in */sys/c/c*, and, if no errors occur, moves them to */bin* and */lib*. "Clean" just removes any object files (*.o*) left around. Note that the *make* description file for the C compiler is *makefile* (no leading *.*). Only the *make* description files that live in object directories (such as */bin*) start with *.*.

Often it is better to install a new version of the C compiler as *ncc*, so that it can be tested without destroying the old (working) version. The following creates and installs a complete new version of the C compiler as *ncc*:

```
chdir /sys/c/c
make -f makefile PREF=n install clean
```

As before, this creates *cc*, *c0*, *c1*, *c2*, and *cpp* in */sys/c/c*. But they are installed as *ncc*, *nc0*, *nc1*, *nc2*, and *ncpp* in */bin* and */lib*. This works because the *cc* driver looks at the name under which it was invoked. If the first character of the name is 'c', it calls *c0*, etc., in */lib*. But if the first character isn't 'c', it prepends that character to the names of the load modules that it calls. Thus *ncc* calls *nc0*, *nc1*, *nc2*, and *ncpp* in */lib*.

**/usr/hasp:**

The *.makefile* for */usr/hasp* specifies whether your RJE system supports one or two lines. The following generates a two-line system:

```
chdir /usr/hasp
make -f .makefile NHASP=2
```

while:

```
chdir /usr/hasp
make -f .makefile NHASP=
```

generates a one-line system (the default is one-line). To change the default, change the "NHASP=" line near the beginning of the *.makefile*.

**SEE ALSO**

*make(I)*

*Make - A Program for Maintaining Computer Programs* by S. I. Feldman

**DIAGNOSTICS**

"Don't know how to make xxx": The source file xxx doesn't exist.

**FILES**

*.makefile\**

**BUGS**

Not every directory has a *.makefile*.

—

REGEN ( VIII )

PWB/UNIX 5/31/77

REGEN ( VIII )  
—

*Nroff*(I) and *troff*(I) are fixed to Eastern Standard Time; they use their own conversion routines, instead of calling *ctime*(III). Fortunately, they use just the date, not the time-of-day.

