Installation Notes

Cygnus Support Developer's Kit
Release progressive-96q1

Contents:

Cygnus Support hotline: +1 415 903 1401

Installing in brief for Unix systems

You can run the brief installation procedure if:

- You are installing on a standard Unix platform (see Appendix A "Platform Names," page 27)
- Your Unix machine has its own device which corresponds to your distribution media (e.g., a QIC-24 tape drive)
- You're willing to use the installation directory '/usr/cygnus'
- You have enough disk space in '/usr/cygnus' (your tape label lists the required disk space for binary, source, and both)

Otherwise, see "Developer's Kit installation on Unix," page 7.

Steps for Brief Install:

(In examples, we show the system prompt as 'eg$'.)

1. Make sure you can write in '/usr/cygnus' by typing:

```
eg$ su root                 (enter root password)
# mkdir /usr/cygnus         (ignore "File exists" error if any)
# chmod 777 /usr/cygnus
# exit                      (root access not needed beyond this)
```

2. Load the distribution into the drive and extract the `Install` script. **WARNING:** you must use a *non–rewinding* tape device; see "Device names," page 5.

```
eg$ cd /tmp
eg$ tar xfv device Install
```

3. Run the `Install` script:

```
eg$ ./Install -tape=device
```

   `Install` displays messages about its activity, ending with

```
Done.
```

4. Build symbolic links to make execution paths easy (you may need root access to put the link in '/usr'):

```
eg$ cd /usr/cygnus
eg$ ln -s progressive-96q1 progressive
eg$ su root
# ln -s /usr/cygnus/progressive/H-host /usr/progressive
# exit              (give up root access as soon as possible)
```

5. Use your Cygnus Support customer ID (see cover letter) to tag your copy of our problem-report form:

```
eg$ /usr/progressive/bin/install-sid customer-ID
```

6. Remove public write access from '/usr/cygnus'. See your system administrator for the correct permissions at your site.

You're done! Anyone who puts '/usr/progressive/bin' in their PATH can use this Developer's Kit distribution.

We have attempted to make the installation of the Cygnus Support Developer's Kit distribution as trouble-free as possible. If you encounter any problems, please contact us:

Cygnus Support
toll free: +1 800 CYGNUS-1
main line: +1 415 903 1400
hotline: +1 415 903 1401
email: support@cygnus.com

Headquarters
1937 Landings Drive
Mountain View, CA 94043 USA

+1 415 903 1400
+1 415 903 0122 fax

East Coast
48 Grove St., Ste. 105
Somerville, MA 02144 USA

+1 617 629 3000
fax +1 617 629 3010

Faxes are answered 8 am–5pm, Monday through Friday.

Installing in brief for MS-DOS systems

All MS-DOS releases of the Developer's Kit can use this brief installation procedure. For more detail on the installation procedure, see "Developer's Kit installation on MS-DOS," page 21.

We ship your Developer's Kit on a set of floppy disks. The `INSTALL` program is included on Disk 1.

Steps for Brief Install:

1. Insert Disk 1 into the floppy drive (the example shows '`A:`') and type:
   ```
   A:\INSTALL
   ```
2. `INSTALL` prompts for an installation directory; the default is '`C:\CYGNUS`'. The tools may be installed anywhere. (If you are installing more than one Developer's Kit, see "MS-DOS Installation Directories," page 21.)

   Make sure the installation directory is correct and press `RETURN`.
3. `INSTALL` first checks the installation location to make sure it has enough space before unpacking the tools. Installed Developer's Kit disk usage varies from about 10 to about 16 megabytes, depending on the target. The disk labels list the required disk space for each distribution.
4. `INSTALL` reads the first disk and then requests the next.
5. After the last disk, press `RETURN` to exit the `INSTALL` program.
6. To run the programs, type '*`installdir`*`\SETENV`' to set up your working environment. (*`installdir`* is the installation directory you specified, '`C:\CYGNUS`' by default.)
   ```
   C:\> cd c:\cygnus
   C:\CYGNUS\> setenv
   . . .
   ```
7. To test the installation, change your working directory to the '*`installdir`*`\DEMO`' subdirectory and type '`MAKE`':
   ```
   C:\CYGNUS\> cd c:\cygnus\demo
   C:\CYGNUS\DEMO> make
   . . .
   ```
8. It is often easiest to set your working environment in the initialization file '`AUTOEXEC.BAT`'. If you installed in the default '`C:\CYGNUS`', you can simply add the following line to your '`AUTOEXEC.BAT`':
   ```
   CALL C:\CYGNUS\SETENV.BAT
   ```

Developer's Kit installation on Unix

This section describes host-specific information and installation instructions for Unix systems.

Device names

For Unix distributions, your distribution tape includes two files:

`Install` The `Install` shell script is a portable installation procedure which automatically installs the software on your system (see "Invoking the `Install` script," page 8).

distribution `tar` *file*
The binaries and source for the Developer's Kit distribution are located in a single compressed `tar` file.

In order for `Install` to read and install your distribution, you *must* use a *non–rewinding* tape device, so that the tape drive maintains the tape location at the beginning of the compressed `tar` file after you extract `Install`.

These are examples of non–rewinding tape devices for each system; see your system administrator for the name of the non–rewinding tape device on your particular host. Also shown are MAN pages to which you can refer for more information about tape devices.

Standard QIC-24 tape drives:

platform	*device*	*man page*
`sparc-sun-solaris2`	`/dev/rmt/0n`	*use* `man st`
`sparc-sun-sunos4.1.3`	`/dev/nrst8`	*use* `man st`
`mips-dec-ultrix`	`/dev/nrtm0`	*use* `man mtio`
`mips-sgi-irix4`	`/dev/mt/tps0d0nrns`	*use* `man tps`
(Note: You must also use a non-byte-swapping device for the SGI Iris)		
`rs6000-ibm-aix`	`/dev/rmt0.1`	*use* `man rmt`
`i386-sysv4.2`	`/dev/rmt/c0s0n`	*use* `man tape`

Standard DAT tape drives:

platform	*device*	*man page*
`m68k-hp-hpux`	`/dev/rmt/0mn`	*use* `man 7 mt`
`hppa1.1-hp-hpux`	`/dev/rmt/0mn`	*use* `man 7 mt`
`mips-sgi-irix5`	`/dev/mt/tps1d5nrns`	*use* `man tps`
(Note: You must also use a non-byte-swapping device for the SGI Iris)		

Disk space requirements

This section lists the minimum disk space requirements (in megabytes) for installations of binaries only, source code only, or the sum total of both. For more information on binary- or source-only installations, see "Developer's Kit installation on Unix," page 7. Sizes shown are for *native* distributions; see your tape label for the actual disk size.

platform	*binaries*	*source*	*total*
sparc-sun-solaris2	44	71	115
sparc-sun-sunos4.1.3	31	71	102
mips-dec-ultrix	39	71	110
rs6000-ibm-aix	44	71	115
mips-sgi-irix4	39	71	110
m68k-hp-hpux	32	71	103
hppa1.1-hp-hpux	46	71	117
i386-sysv4.2	48	71	119
alpha-dec-osf1.3	56	71	127

Operating System requirements

This section lists the minimal operating system requirements for each Unix system.

platform	*OS level*
sparc-sun-solaris2	Solaris 2.*x*
sparc-sun-sunos4.1.3	SunOS 4.1.*x*
mips-dec-ultrix	Ultrix 4.2
rs6000-ibm-aix	AIX 3.2
mips-sgi-irix4	Irix 4.*x*
m68k-hp-hpux	HP/UX 8.*x*
hppa1.1-hp-hpux	HP/UX 8.*x*
i386-sysv4.2	UnixWare SysVr4.2, version 1.1.1
alpha-dec-osf1.3	OSF/1 1.3

Installing your Developer's Kit distribution

There are a few steps to follow in installing the software in the Developer's Kit distribution onto your system.

Note: For Unix distributions, your distribution tape includes two files:

`Install` The `Install` shell script is a portable installation procedure which automatically installs the software on your system (see "Invoking the `Install` script," page 8).

distribution `tar` *file*
The binaries and source for the Developer's Kit distribution are located in a single compressed `tar` file.

In order for `Install` to read and install your distribution, you *must* use a *non–rewinding* tape device, so that the tape drive maintains the tape location at the beginning of the compressed `tar` file after you extract `Install`. See "Device names," page 5, for a list of default device names for each host type.

1. First, decide where to install the software. The default installation location is '`/usr/cygnus/progressive-96q1`'. (To use the software conveniently from elsewhere, you may want to reconfigure and recompile from source; see "Running the programs," page 16.)

   If you don't wish to install in '`/usr/cygnus`' but can create a symbolic link to it from another location, or if you don't wish to install into '`/usr`' at all, see "Installing in a nonstandard location," page 14.

2. Create the installation directory, if it doesn't already exist, and make sure it's publicly accessible so `Install` can write there. For example, if you use the default installation directory of '`/usr/cygnus`':

   ```
   eg$ su root              (enter root password to write in '/usr')
   # mkdir /usr/cygnus      (ignore "File exists" error if any)
   # chmod 777 /usr/cygnus
   # exit                   (root access not needed beyond this)
   ```

3. Make sure you have enough space for the tools in your chosen installation location. The required disk usage for the Developer's Kit is printed on the tape label; values for binaries only, sources only, or both are shown.

4. Load the Developer's Kit distribution tape into your tape drive. If your machine doesn't have its own tape drive, you need to first extract the software into a location accessible by both your host and the machine that has a tape drive, and then install on your host. If there is no shared disk, you can extract the software on the machine with the tape drive and then transfer it over to your host. "Installing with a remote tape drive," page 14, for details.

5. Extract the `Install` script off the tape using

```
tar xvf non-rewinding-tapedev Install
```

Remember to use a *non–rewinding* tape drive!

6. Run `Install`, using command-line options and arguments to specify the details about your installation.

   Default behavior installs both binaries and source under '`/usr/cygnus/progressive-96q1`' using the default non–rewinding tape drive for your system (see "Device names," page 5). For *native* toolchains only, a process called *fixincludes* automatically makes copies of your system header files and alters them to work with GCC (your system's header files are *not changed*; see "Why convert system header files?," page 11). Finally, `Install` runs a simple test to make sure your distribution was installed correctly.

7. Make sure the program `send-pr` knows your Cygnus customer identification code. You can install your customer ID by using the program `install-sid` as follows:

   ```
   eg$ cd /usr/cygnus/progressive-96q1/H-hosttype/bin
   eg$ install-sid customer-ID
   ```

   Contact Cygnus Support at +1 415 903 1401 if you do not know your customer ID.

8. Create symbolic links so that your newly installed Developer's Kit is easily accessible to developers, able to exist with other Developer's Kit installations in a heterogeneous environment, and easily updated when you install a new Developer's Kit.

   The nature of the links depends on where you installed the Developer's Kit release, but they follow the example below. If you installed into '`/usr/cygnus/progressive-96q1`', the links are

   ```
   ln -s /usr/cygnus/progressive-96q1 /usr/cygnus/progressive
   ln -s /usr/cygnus/progressive/H-hosttype /usr/progressive
   ```

   See "Links for easy access and updating," page 12, for more information on these links.

   You're done! The installation is now online; anyone who puts

   ```
   /usr/progressive/bin
   ```

   in their path has access to the toolkit.

9. If you had to change the permission status on the directory '`/usr/cygnus`', be sure to revert the change. See your system administrator for the proper permissions at your site.

Invoking the `Install` script

There are two kinds of command-line arguments to `Install`, which you can use to direct its operation:

- *What form of the programs* to install. You can choose between binaries (argument `bin`) and source code (`source`). If you don't specify either of these, `Install` assumes you want *both source and binaries*.
- *What installation actions* to carry out. A full installation involves up to three steps; `Install` has options to let you choose them explicitly. The steps are

  1. extracting source from the tape (option `extract`)
  2. writing copies of your system '`include`' files, adjusted for portability (needed for the compilation tools; option `fixincludes`)
  3. running a simple test of the installed programs (option `test`)

  The last two of these actions (`fixincludes` and `test`) are *not needed for cross-development* configurations. (A cross-development configuration runs on a *host*, but is meant to develop code for another platform, the *target*. Cross development tapes have '`target = target`' on the tape label.)

  These two actions can only run on your host. If you read the tape on another machine, you must specify the `extract` option explicitly, to indicate that you don't expect the other two actions to run (and are aware of the need to run further installation steps on your host).

`Install` also has two command line options: '`-tape`' and '`-installdir`'. You can use these to adapt the installation to your system.

`Install` options

```
Install [ bin ] [ source ]
        [ extract ] [ fixincludes ] [ test ]
        [ -tape=device ]
        [ -installdir=directory ]
```

`bin`
`source` By default, `Install` extracts both source and binaries. Instead of relying on the default, you can use these options to specify exactly what you want. You need to do this if you want *only* binaries or *only* source.

`Install` is designed to share files, wherever possible, between installations for different hosts (of the same release). If you get Cygnus release tapes configured for different hosts, there is no need to do a binary-only install of some of the tapes to save space on a shared file system; `Install` arranges the files so that all hosts share the same source files. Documentation files are shared as well. Note that it is faster to extract

the source code only once if you are installing the Developer's Kit distribution for more than one host.

See "Links for easy access and updating," page 12, for a discussion of how to manage the directory structure used for this purpose.

`extract`
`fixincludes`
`test`

In a cross-development configuration, only the '`extract`' step is used.

In a native configuration—meant for developing software on the same host where the Developer's Kit runs—a full installation includes up to three things: (1) extracting software from the tape; (2) creating ANSI-C conforming copies of your system's standard header files; and (3) testing the installation. You can execute these steps separately by specifying '`extract`', '`fixincludes`', or '`test`' on the `Install` command line.

In the native configuration, after you run '`extract`', '`fixincludes`' is essential to the compiler. '`fixincludes`' *does not change your system's original header files;* `Install` writes the converted copies in a separate, gcc-specific directory. See "Why convert system header files?," page 11, for more discussion of the '`fixincludes`' step. `Install` only attempts these last two steps if you run it on the host for which the binaries were compiled.

When you run '`extract`', `Install` creates a log file in '`/usr/cygnus/progressive-96q1/extraction.log`'. When you run '`fixincludes`', `Install` creates a log file in '`/usr/cygnus/progressive-96q1/fixincludes.log`'.

'`test`' (used only for the native configuration) is a confidence-building step, and doesn't actually change the state of the installed software. The '`test`' step may not make sense, depending on what other options you've specified—if you install only source, there's nothing to test.

`-tape=`*`device`*
`-tape=`*`tarfile`*

Specify the *non–rewinding* device name for your tape drive as *`tape`*.

If you extract the installation script and *`tarfile`* on some other system, and transfer them to your host for installation, use the name of the `tar` file instead of a device name with

'`-tape`'. See "Installing with a remote tape drive," page 14, for more discussion.

`-installdir=`*`directory`*
: If you cannot or do not wish to install into '`/usr/cygnus`', use this option to specify an alternate *`directory`* for placing your software—but beware: the software is configured to go in '`/usr/cygnus`', and you'll have to override or change that too. See "Running the programs," page 16.

If you specify a step that doesn't make sense, `Install` notices the error, and exits (before doing anything at all) with an error message, so you can try again.

Why convert system header files?

> The '`fixincludes`' installation step described here *applies only to the native configuration* of the Developer's Kit—that is, only if your tape is configured to develop software for the same *host* on which it runs. If you have a cross-development tape, configured to develop software for another machine (the *target*), the system header files from your *host* are not needed for the GNU compilers. Cross-development tapes have '`target = `*`target`*' on the tape label.

For the native configuration, it is very important to run '`Install fixincludes`' (on *each host* where you install the compiler binaries).

When the ANSI X3J11 committee finished developing a standard for the C language, a few things that had worked one way in many traditional C compilers ended up working differently in ANSI C. Most of these changes are improvements. But some Unix header files still rely on the old C meanings, in cases where the Unix vendor has not yet converted to using an ANSI C compiler for the operating system itself. '`Install fixincludes`' does a mechanical translation that writes ANSI C versions of some system header files into a new, GCC-specific include directory—*your system's original header files are not affected.*

The C header files supplied with SVr4 versions of Unix depend on a questionable interpretation of the ANSI C standard: they test for a non-ANSI environment by checking whether `__STDC__` is defined as zero. The ANSI standard actually only specifies that `__STDC__` be defined to 1; if it is defined to any other value, the environment is not ANSI C compatible, and ANSI C says nothing about what that value might be.

GCC defines `__STDC__` to 1 when running with '`-ansi`', when it functions as an "ANSI C superset" compiler. (It also sets `__STRICT_ANSI__`

when it runs with the '`-pedantic`' option.) However, GCC leaves `__STDC__` undefined when it is not running as an ANSI C compiler.

Unfortunately for Solaris users, Solaris header files follow the SVr4 choice. Since GCC never defines `__STDC__` as 0, the distributed header files can leave out some declarations. (Look in '`/usr/include/time.h`', for example.)

Part of the installation process of the native compiler release is to "fix" the header files, such as '`stdio.h`', on the host system to remove ANSI incompatibilities. '`Install fixincludes`' makes copies of the system '`include`' files which have these nonstandard features removed, so that GCC can process them. These copies are placed in a new, GCC-specific '`include`' directory—*your system's original header files are not affected*. Once these fixed header files are created, GCC finds and uses them automatically.

Likewise, C++ programmers require C++-ready, ANSI-compatible versions of the standard C header files. These used to be provided with `libg++`, but were difficult to maintain due to the design compromises (and outright "kludges") that were necessary to make these work on all the systems we support.

We have recently introduced what we believe to be a better solution in the form of a new shell script, `fixproto`. `fixproto` analyzes all the header files in '`/usr/include`', and adds any missing standard ANSI and Posix.2 prototypes. The '`extern "C"`' braces needed to specify that these are C (not C++) functions are also added as needed. It is run as part of the installation and/or build of a native compiler. The resulting header files are also used for C, with the result that the '`-Wimplicit`' option for `gcc` is much more useful.

The most obvious drawback to this solution is that the process of "fixing" the '`include`' files takes longer to run, so any installation of a native compiler is noticeably slower than in previous releases. Performance improvements will be made as part of a future release.

If you don't run `fixincludes`, the GNU C compiler can only use the original system header files when you compile new C programs. *In some cases, the resulting programs will fail at run-time.*

Links for easy access and updating

Once you've extracted the tools from the tape, they are installed into a directory named '*`installdir`*`/progressive-96q1`'. We put the release number in the directory name so that you can keep several releases installed at the same time, if you wish. In order to simplify administrative procedures (such as upgrades to future Cygnus Support Pro-

gressive releases), we recommend that you establish a symbolic link '/usr/cygnus/progressive' to this directory.

```
ln -s installdir/progressive-96q1 installdir/progressive
```

For example, if you've installed in the default location under '/usr/cygnus':

```
ln -s /usr/cygnus/progressive-96q1 /usr/cygnus/progressive
```

Directories of *machine-independent* files (source code and documentation) are installed directly under 'progressive-96q1'. However, to accomodate binaries for multiple hosts in a single directory structure, the binary files for your particular host type are in a subdirectory 'H-*hosttype*'. (*hosttype* indicates a particular architecture, vendor and operating system. See Appendix A "Platform names," page 27.)

This means that one more level of symbolic links is helpful, to allow your users to keep the same execution path defined even if they sometimes use binaries for one machine and sometimes for another. Even if this doesn't apply now, you might want it in the future; establishing these links now can save your users the trouble of changing all their paths later. The idea is to build '/usr/progressive/bin' on each machine so that it points to the appropriate binary subdirectory for each machine—for instance, '/usr/cygnus/progressive/H-*hosttype*'.

You may need super-user access again briefly to establish this link:

```
ln -s /usr/cygnus/progressive/H-hosttype /usr/progressive
```

We recommend building these links as the last step in the installation process. That way, users at your site only see software in '/usr/progressive' when you're satisfied that the installation is complete and successful.

Installation variances

Once you've extracted `Install` from your tape, you can tell `Install` what software to install, what form of the programs you need, and what installation steps to do. Here are some examples covering common situations. For a full explanation of each possible `Install` argument, see "Invoking the `Install` script," page 8.

`Install`'s default tape drive is the non-rewinding tape drive for your system (see "Device names," page 5), which is right for the most common cases. If your tape drive is different, you need to use the '-tape=/dev/*tape*' option; the examples show this option for completeness. Remember to specify a *non–rewinding* tape device.

Installing only binaries or source

If you don't want the source—for instance, to save space—you can use the argument 'bin'.

```
eg$ tar xvf device Install
Install
eg$ ./Install -tape=device bin . . .
```

By the same token, if you don't wish to install the binaries—for instance, if you plan to rebuild them from source anyway—you can use the argument 'source'.

```
eg$ tar xvf device Install
Install
eg$ ./Install -tape=device source . . .
```

Installing in a nonstandard location

If you wish to install this Developer's Kit distribution in a directory other than the default, '/usr/cygnus', use the '-installdir' option to Install. Remember, though, you must set some environment variables in order for the tools to function at all. See "Running the programs," page 16.

```
eg$ cd /tmp
eg$ tar xvf device Install
Install
eg$ ./Install -tape=device -installdir=somewhere bin
. . .
```

Installing with a remote tape drive

If your host doesn't have an appropriate tape drive, you may still be able to install your software. Check with your system administrator to see if another machine at your site has a tape drive you can use. If so:

If a shared filesystem is available
between the two machines, and it has enough space, create '/usr/cygnus' on your host (the one where you want to install this Progressive Release) as a symbolic link to a directory where the other machine (the one with a tape drive) can write:

```
ln -s shared /usr/cygnus
```

Run Install from the machine with a tape drive, using the 'extract' argument and the '-installdir' option:

```
Install extract -installdir=shared
```

You still have to finish the installation, but the last two steps (fixincludes and test) must be run on your host. (If you

forget, there's no great harm done: `Install` notices that it can't carry out a full installation on the wrong machine, and stops with an error message—then you can go back and try again. When `Install` notices a problem like this, it doesn't carry out *any* action other than giving a helpful error message).

Unless you are installing a cross-development tape (the tape label says '`target = target`' for cross configurations), the '`fixincludes`' part of the installation is essential. Please see the full explanation (see "Why convert system header files?," page 11), if you're curious.

On a machine on your network with a tape drive:
```
./Install extract -installdir=shared/cygnus ...
```

On your host
```
ln -s shared/cygnus /usr/cygnus
cd /usr/cygnus/progressive-96q1
```

If your copy of the Developer's Kit is configured *native* (to develop software for the same type of machine where the Developer's Kit itself runs), you'll have to run '`Install fixincludes`' and '`Install test`' from your host afterwards.

Native configurations only:
```
./Install fixincludes test
```

If some form of filetransfer is available

(such as uucp), read the second file on the tape using a system utility (for instance, `dd` on Unix systems; see the system documentation for the machine with a tape drive). There are two files on the distribution tape; the first contains just the `Install` script in uncompressed `tar` format, and the second is a compressed `tar` format file containing the rest of the release.

Read both of these files separately, using something like the following:

```
eg$ tar xvf non-rewinding-tape-device Install
Install
eg$ dd if=non-rewinding-tape-device of=tarfile1 bs=62k
messages from dd...
eg$ ls
Install
tarfile1
```

and then transfer them to your own machine using `uucp`, `ftp`, or another appropriate file transfer tool. (The blocksize is set to 62k in this example simply to speed up the process; the tape is written with a blocksize of 62k, but `dd` should be able to cope with the task using its default blocksize.)

Then run `Install`, but use '`-tape=`*`tarfile`*' to specify the name of the installation file, instead of '`-tape=`*`device`*' as shown in the examples. In the simplest case, for example (starting after you've transferred `Install` and the tar file to your system):

```
eg$ ./Install -tape=tarfile1
```

Running the programs

In order to run the tools in the Developer's Kit release after you install them, you must first set a few environment variables so your shell can find them.

- At the very least, you must set your `PATH` variable. See "Setting PATH," page 16.
- If you installed the tools in a location other than the default and choose not to set the standard symbolic links in place (see "Links for easy access and updating," page 12), you must also set the environment variable `GCC_EXEC_PREFIX`. Otherwise, the compiler cannot find its resources. See "GCC paths," page 17.
- If you install the Developer's Kit tools in an alternate location, you need to set the variable `INFOPATH` so that `info` can find the online documentation. See "Online documentation paths," page 17.
- Some `man` programs recognize the environment variable `MANPATH` as a search path for online manual pages. You must either add your installation directory to your `MANPATH` environment variable, or copy the online manual pages in your distribution into a location where your `man` program can find them. See "Online documentation paths," page 17.

Setting PATH

Any user who wishes to run the tools in this distribution needs to make sure her `PATH` environment variable can find the tools. Whether you install in the default location:

```
/usr/cygnus/progressive-96q1
```

or in an alternate location, you need to alter your `PATH` environment variable to point toward the newly installed tools.

If you create the symbolic links we recommend (see "Links for easy access and updating," page 12), users who want to run the Developer's Kit—regardless of whether they need binaries for your particular host, or for some other platform—can use settings like one of the following in their initialization files.

This example shows '/usr/progressive/bin' as the final linked installation directory. If you installed into a directory other than this, substitute the actual directory for '/usr/progressive/bin'.

For Bourne-compatible shells (/bin/sh, bash, *or Korn shell):*

```
PATH=/usr/progressive/bin:$PATH
export PATH
```

For C shell:

```
set path=(/usr/progressive/bin $path)
```

GCC paths

You can run the compiler GCC without recompiling, even if you install the distribution in an alternate location, by first setting the environment variable GCC_EXEC_PREFIX. This variable specifies where to find the executables, libraries, and data files used by the compiler. Its value will be different depending on which set of binaries you need to run. For example, if you install the tape distribution under '/local' (instead of the default '/usr/cygnus'), and you wish to run GCC as a native compiler, you could set GCC_EXEC_PREFIX as follows.

(*Note:* The sample shows a GCC_EXEC_PREFIX which is split across two lines only to fit on the printed page; it is meant to be typed on one line.)

For shells compatible with Bourne shell (/bin/sh, bash, *or Korn shell):*

```
eg$ GCC_EXEC_PREFIX=/local/progressive-96q1/\
       H-hosttype/lib/gcc-lib/
eg$ export GCC_EXEC_PREFIX
```

For C shell:

```
eg% setenv GCC_EXEC_PREFIX /local/progressive-96q1/\
       H-hosttype/lib/gcc-lib/
```

Note: The trailing slash '/' is important. The gcc program uses GCC_EXEC_PREFIX simply as a prefix. If you omit the slash (or make any other mistakes in specifying the prefix), gcc fails with a message beginning 'installation problem, cannot exec. . .'.

Online documentation paths

The standalone documentation browser info needs to know the location of the documentation files in the distribution. The default location, '/usr/cygnus/progressive-96q1/info', is compiled into info. If you install elsewhere, set the environment variable INFOPATH to indicate the alternate location.

For example, assuming you installed under '/local':

For shells compatible with Bourne shell (`/bin/sh`*,* `bash`*, or Korn shell):*

```
eg$ INFOPATH=/local/progressive-96q1/info
eg$ export INFOPATH
```

For C shell:

```
eg% setenv INFOPATH /local/progressive-96q1/info
```

If you built '`progressive`' as a symbolic link to '`progressive-96q1`', as recommended in "Links for easy access and updating," page 12, then you could simply use '`/local/progressive/info`' as the value of `INFOPATH` in the examples above.

You should also ensure that your `man` command can pick up the manual pages for these tools. Some `man` programs recognize a `MANPATH` environment variable. If your `man` program is one of these, users at your site can also include in their initialization file lines like

For Bourne-compatible shells:

```
eg$ MANPATH=/usr/cygnus/progressive/man:$MANPATH:/usr/man
eg$ export MANPATH
```

For C shell:

```
eg% setenv MANPATH /usr/cygnus/progressive/man:$MANPATH:/usr/man
```

If your `man` program doesn't recognize `MANPATH`, you may want to copy or link the files from '*`installdir`*`/progressive/man/man1`' into your system's '`man/man1`' directory.

Some Things that Might go Wrong

We've tried to make the installation of the Developer's Kit distribution as painless as possible. Still, some complications may arise. Here are suggestions for dealing with some of them.

No customer ID for `send-pr`

Make sure the program `send-pr` knows your Cygnus customer identification code. You can install your customer ID by using the program `install-sid` as follows:

```
install-sid customer-ID
```

If you installed the Developer's Kit into a location other than the default, and you chose not to set up symbolic links pointing to the real installation location, you need to use the '`--install-dir`' option to `install-sid` as follows:

```
install-sid --install-dir=install-dir-prefix customer-ID
```

where *install-dir-prefix* points to the top level of the installation. Contact Cygnus Support at +1 415 903 1401 if you do not know your customer ID.

Not enough space

If you don't have enough space to install all of the tape distribution, you can instead extract only the compiled code, or only the source.

You can easily extract these components independently of one another by using the 'source' or 'bin' arguments to `Install`. See "Invoking the `Install` script," page 8.

No access to '/usr/cygnus'

If you can't sign on to an account with access to write in '/usr' or '/usr/cygnus', use the '`-installdir=`*`directory`*' option to `Install` to specify a different installation directory to which you *can* write. For example, if all the other installation defaults are right, you can execute something like '`./Install -tape=/dev/`*`tape`* `-installdir=`*`mydir`*'. You'll need to either override default paths for the pre-compiled tools, or else recompile the software. See "Running the programs," page 16, and "Links for easy access and updating," page 12, for details.

> *WARNING:* If you can't install in '/usr/cygnus' (or link your installation directory to that name), some of the defaults configured into the `progressive-96q1` distribution won't work. See "Running the programs," page 16, for information on overriding or reconfiguring these defaults.

Error messages from `Install`

The `Install` script checks for many errors and inconsistencies in the way its arguments are used. The messages are meant to be self-explanatory. Here is a list of a few messages where further information might be useful:

`Can not read from TAPE device,` *`tape`*

> The error message ends with the tape device `Install` was trying to use. Please check that it is the device you intended; possible causes of trouble might include leaving off the '/dev/' prefix at the front of the device name. A typographical error in the device name might also cause this problem.

If the problem is neither of these, perhaps your tape device can't read our tape; see "Installing with a remote tape drive," page 14, for a discussion of how to use another machine's tape drive, or contact Cygnus Support.

`stdin: not in compressed format`
: You are probably not using the non–rewinding tape device. There are two files on each tape. The first is a `tar` file containing the `Install` script. The second is a compressed `tar` file containing everything else. Without using the non–rewinding device, there is no way to skip over the first file to begin reading the second.

`gcc: cannot exec cpp`
: If you've installed the binary distribution of the Developer's Kit software in a non-standard location, remember to set your environment variable `GCC_EXEC_PREFIX` accordingly. See "Running the programs," page 16.

`. . . This is a problem.`
`Cannot cd to` *`installdir`*
`I do not know why I cannot create` *`installdir`*
`hello.c fails to run`
`test-ioctl.c fails to run`
: These errors (the first covers anything that ends in '`This is a problem`') are from paranoia checks; they are issued for situations that other checks should have covered, or for unlikely situations that require further diagnosis.

  If you get one of these messages, please call the Cygnus hotline, +1 415 903 1401, or send electronic mail to '`help@cygnus.com`'.

Developer's Kit installation on MS-DOS

This section describes the installation procedure for the Cygnus Support Developer's Kit distribution running on MS-DOS. For specific information about using this release with MS-DOS, see the MS-DOS specific developer's note, *Developing with DOS*.

MS-DOS installation directories

You may install the software in any directory. The `INSTALL` program assumes that you are installing in `C:\CYGNUS`; if you want to change this, you may do so at the beginning of the installation process.

If you are installing cross-development tools for more than one target, you *must* install them into different directories; otherwise, the second installation overwrites the first. We recommend you use a directory structure that has the target name built-in, such as:

```
C:\CYGNUS\A29KUDI\. . .     AMD 29k cross-development toolkit
C:\CYGNUS\M68KCOFF\. . .    Motorola 68k cross-development toolkit
. . .
```

If you use this paradigm, remember to type

disk`:\CYGNUS\`*target*`\SETENV`

to reset your environment every time you switch targets.

Disk space

The total space required to extract and install binaries for all programs in the Developer's Kit is from 10 to 16 megabytes, depending on the target. The actual disk space required by the Developer's Kit is printed on the disk label. The `INSTALL` program dynamically compares the space available on your designated drive with the size of the installation before starting the installation. If you do not have enough space to install the binaries, the `INSTALL` program exits with an error message before writing anything.

Note that if you're using a disk compression utility like `STACKER` then the actual amount of disk space that you have available may be less than reported. This is because the compression utilities usually report the amount of free space available assuming that the data which would go into it would be compressed at least as well as the data already on the disk. This is important information if you're trying to install the tools onto a compressed disk with only just enough room for the installation. It could be that you run out of real disk space before the installation is complete because the compression utility couldn't do as good a job as it expected.

MS-DOS memory requirements

We do not recommend using the cross-development kit with less than four (4) megabytes of RAM.

We provide a MS-DOS extender with the cross-development kit for MS-DOS which swaps programs to disk when MS-DOS runs out of memory. To avoid excessive swapping you must have at least 2 megabytes of RAM to run G++ on a PC with MS-DOS.

If you've got more than 2 megabytes, the extra memory can be used as a disk cache to significantly improve performance.

Installation your Developer's Kit

We ship your Developer's Kit on a set of floppy disks. The `INSTALL` program is included on Disk 1.

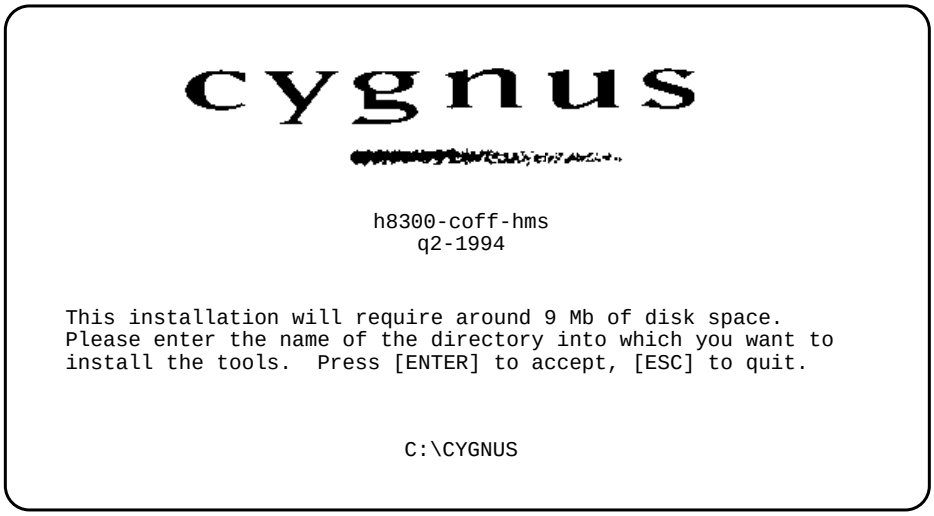
The files are stored on the floppies using Microsoft's `COMPRESS` program. If you prefer, you can install files without using the `INSTALL` program by just copying them into the right place on your hard drive and running `EXPAND` on each file. Since the files are stored on the floppy using their full name (not those marked as compressed by using Microsoft's '`.XX_`' naming convention) you *must* use a temporary file.

> *Warning!* If you have a program in your path called `EXPAND` and it's not the one provided by Microsoft, then you should either change your path to use only the Microsoft `EXPAND` program, or be certain that your `EXPAND` program can decompress files which have been compressed using Microsoft `COMPRESS`.

We show the system prompt as '`C:\>`' for the local hard disk drive, and '`A:\>`' for the local 3.5" floppy disk drive.

For these examples we assume that you install into a directory called '`C:\CYGNUS`' and that you use drive '`A:`' to read the installation floppies. Substitute other hard drives, installation directories, and floppy drives to match your environment.

The `INSTALL` program first prompts you for an installation directory:

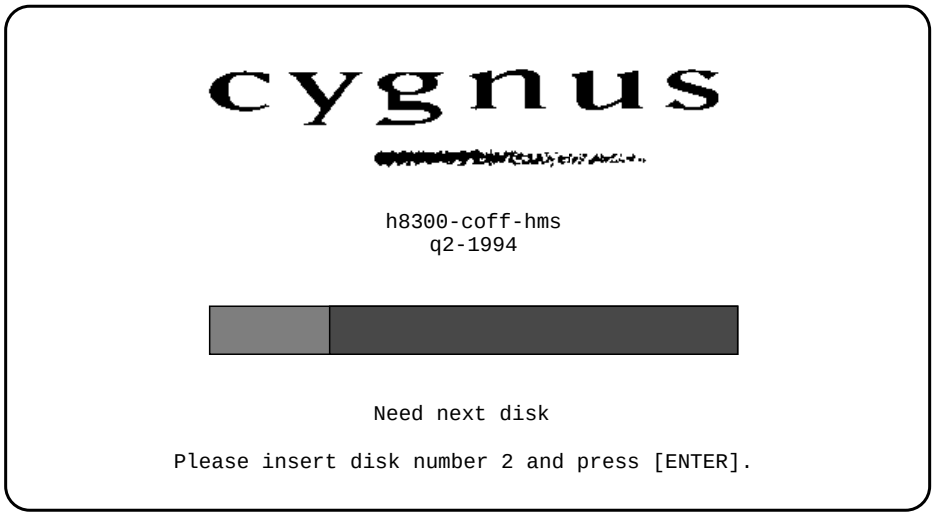


At this prompt, enter the name of the directory where you want the tools to be installed and press ENTER. INSTALL prompts you (assuming that you accept 'C:\CYGNUS'):

```
The installation will write into C:\CYGNUS.
Are you sure you want to continue [Y] or [N]
```

If you type N, INSTALL asks for another path name. If you type Y, INSTALL begins the installation.

The program draws a status bar, which fills up as INSTALL works. When the box is full, the installation is complete. INSTALL shows the name of the file being processed at the bottom of the screen.



INSTALL prompts you for each disk in order.

To use another installation directory, specify the path to your desired directory wherever the examples show 'C:\CYGNUS'

You must execute the batch file 'SETENV.BAT' before running the Developer's Kit. You can set the environment automatically whenever you boot the machine by putting the following line in your 'AUTOEXEC.BAT':

```
CALL C:\CYGNUS\SETENV.BAT
```

(If you install in a location other than 'C:\CYGNUS', be sure to specify the correct directory.)

Release contents

The programs in this Developer's Kit are shipped as binaries, preconfigured to run on Intel *x*86 PCs running standard MS-DOS.

The individual programs in the Developer's Kit are:

REL	Name of the release
MANIFEST	Disk contents manifest
README\COPYING	Information about copying this release
README\README	Last minute information
BIN\AR.EXE	Archive utility
BIN\AS.EXE	Assembler
BIN\ASYNCTSR.COM	Serial line driver TSR
BIN\CC1.EXE	C compiler
BIN\CC1PLUS.EXE	C++ compiler
BIN\CPP.EXE	C preprocessor
BIN\CXX.EXE	C++ compiler driver
BIN\CXXFILT.EXE	C++ symbol name filter
BIN\EMU387	387 emulator
BIN\GASP.EXE	Assembler preprocessor
BIN\GCC.EXE	C compiler driver
BIN\GDB.EXE	Debugger
BIN\GO32.EXE	dos extender
BIN\GXX.EXE	C++ compiler driver
BIN\INFO.EXE	Documentation browser
BIN\LD.EXE	Linker
BIN\MAKE.EXE	Recompilation director
BIN\NM.EXE	Symbol name utility
BIN\OBJCOPY.EXE	Object file copier and converter
BIN\OBJDUMP.EXE	Object file dumper
BIN\RANLIB.EXE	Archive indexer
BIN\SIZE.EXE	Object file size utility
BIN\STRINGS.EXE	Object file strings utility
BIN\STRIP.EXE	Object file symbol stripper
DEMO\HELLO.C	Demonstration program
DEMO\INIT.BAT	Demonstration initialzation batch file
DEMO\MAKEFILE	Makefile for demonstration
INSTALL.EXE	The install program
LIB\LIBC.A	ansi C library
LIB\LIBGCC.A	Compiler support library
LIB\LIBM.A	Maths library

`INCLUDE\_ANSI.H` Include files for C library
`INFO\*.INF` Online documentation, read with `info.exe`
`LIB\LDSCRIPT\ELF32MIP.`*x* Linker information scripts

The included libraries and some utilities are different depending on which target this release is intended for. The file '`MANIFEST`' in the root directory of the first installation disk contains a complete list of everything included in this release.

How to report bugs

If you find a bug in this release, please report it to Cygnus Support.

Use a copy of the Cygnus bug-report form to ensure that we can respond to your bug as quickly as possible. The file '`SENDPR.TXT`' in the installation directory contains a blank copy of this form. To save time, customize this form ahead of time with your Cygnus customer ID, as described in the previous section.

The easiest way to report a bug is to fill in a copy of this form on your computer and send it via Internet electronic mail to '`bugs@cygnus.com`'. Otherwise, you can print the file '`SENDPR.TXT`', fill it in, and FAX the problem report to Cygnus at +1 415 903 0122 (Mountain View, California) or +1 617 629 3010 (Somerville, Massachusetts). Contact Cygnus if you have any trouble.

Cygnus Support
hotline: +1 415 903 1401
email: `info@cygnus.com`

Headquarters
1937 Landings Drive
Mountain View, CA 94043 USA

+1 415 903 1400
+1 415 903 0122 fax

East Coast
48 Grove St., Ste. 105
Somerville, MA 02144 USA

+1 617 629 3000
fax +1 617 629 3010

Source code for your Developer's Kit

The QIC-24 tape included with your Developer's Kit contains source code for all the programs. Most DOS systems cannot read this tape; you probably need to find a Unix system to read it.

You need about 71 megabytes of free disk space to hold the source code. To extract the source code into the current working directory on most Unix machines, execute a command like this:

```
dd if=tapedev | compress -d | tar xvf -
```

tapedev stands for the device name for the tape drive. For example, on most Sun workstations, the device name for the QIC-24 tape drive is '/dev/rst8'. Contact your system administrator for the correct tape device for your system.

Please contact Cygnus Support at +1 415 903 1401 if you would like the source code in another form.

Appendix A Platform names

Your tape is labeled to indicate the host (and target, if applicable) for which the binaries in the distribution are configured. The specifications used for hosts and targets in the `configure` script are based on a three-part naming scheme, though the scheme is slightly different between hosts and targets.

Host names

The full naming scheme for hosts encodes three pieces of information in the following pattern:

`architecture-vendor-os`

For example, the full name for a Sun SPARCstation running SunOS 4.1.4 is

`sparc-sun-sunos4.1.4`

Warning: `configure` can represent a very large number of combinations of architecture, vendor, and operating system. There is by no means support for all possible combinations!

The following combinations refer to hosts supported by Cygnus Support. The OS versions shown are the versions under which this release was built. If you have any questions regarding compatibility, please feel free to contact Cygnus Support. (For a matrix which shows all supported host/target combinations, see section “Overview” in *Release Notes*.)

canonical name	*platform*
`sparc-sun-solaris2`	SPARCstation, Solaris 2.4
`sparc-sun-sunos4.1.4`	SPARCstation, SunOS 4.1.4
`mips-dec-ultrix`	DECstation, Ultrix 4.4
`rs6000-ibm-aix3.2.5`	IBM RS/6000, AIX 3.2.5
`mips-sgi-irix4`	SGI Iris, Irix 4.0.5H
`mips-sgi-irix5`	SGI Iris, Irix 5.3
`m68k-hp-hpux9`	HP 9000/300, HPUX B.09.00
`hppa1.1-hp-hpux9`	HP 9000/700, HPUX A.09.05
`hppa1.1-hp-hpux10`	HP 9000/700, HPUX B.10.01
`alpha-dec-osf2.0`	DEC Alpha, OSF/1 v2.0
`powerpc-ibm-aix4.1.3`	IBM PowerPC, AIX 4.1.3

Target names

If you have a cross-development tape, the label also indicates the target for that configuration. The pattern for target names is

`architecture``[-`*`vendor`*`]-`*`objfmt`*

Target names differ slightly from host names in that the last variable indicates the object format rather than the operating system, and the

second variable is often left out (this practice is becoming obsolete; in the future, all configuration names will be made up of three parts).

In cross-development configurations, each tool in the Developer's Kit is installed with the configured name of the target as a prefix. For example, if the C compiler is configured to generate COFF format code for the Motorola 680x0 family, the compiler is installed as '`m68k-coff-gcc`'.

> *Warning:* `configure` can represent a very large number of target name combinations of architecture, vendor, and object format. There is by no means support for all possible combinations!

This is a list of some of the more common targets supported by Cygnus Support. (Not all targets are supported on every host!) The list is not exhaustive; see section "Overview" in *Release Notes*, for an up-to-date matrix which shows the host/target combinations supported by Cygnus. Also, see the manual section "Embed with GNU" in *Embedded Systems Programming*.

Motorola 68000 family

`m68k-aout`	`a.out` object code format
`m68k-coff`	COFF object code format
`m68k-vxworks`	VxWorks environment

Motorola 88000 family

`m88k-coff`	COFF object code format

Intel 960 family

`i960-vxworks5.0`	VxWorks environment (`b.out` format)
`i960-vxworks5.1`	VxWorks environment (COFF format
`i960-nindy-coff`	Nindy monitor

AMD 29000 family

`a29k-amd-udi`	UDI monitor interface
To use the minimon *interface, use this configuration with the auxiliary program* `MONTIP`*, available from AMD.*	
`a29k-vxworks5.1`	VxWorks environment (COFF format)

SPARC family

`sparc-vxworks`	VxWorks environment
`sparc-aout`	`a.out` object code format
`sparclite-aout`	`a.out` object code format
`sparclite-coff`	COFF object code format

Intel 80X86 family

`i386-aout`	`a.out` object code format
`i386-elf`	ELF object code format

IDT/MIPS R3000

`mips-idt-ecoff`	IDT R3000, ECOFF object code format
`mips-elf`	IDT R3000, ELF object code format
`mips64-elf`	IDT R3000, ELF object code format

Hitachi H8300

	`h8300-hms`	COFF object code format
Hitachi SH		
	`sh-hms`	COFF object code format
Z8000		
	`z8k-coff`	COFF object code format

`config.guess`

`config.guess` is a shell script which attempts to deduce the host type from which it is called, using system commands like `uname` if they are available. `config.guess` is remarkably adept at deciphering the proper configuration for your host; if you are building a tree to run on the same host on which you're building it, we recommend *not* specifying the *`hosttype`* argument.

`config.guess` is called by `configure`; you need never run it by hand, unless you're curious about the output.

Appendix B Cross-development environment

Using the Developer's Kit in one of the cross-development configurations usually requires some attention to setting up the target environment. (A cross-development configuration is used for developing software to run on a different machine (the *target*) from the development tools themselves (which run on the *host*)—for example, you might use a SPARCstation to generate and debug code for an AMD 29K-based board.)

To allow our tools to work with your target environment (except for real-time operating systems, which provide full operating system support), you need to set up:

- the C run-time environment (described below).
- *stubs*, or minimal versions of operating system subroutines for the C subroutine library. See section "System Calls" in *The Cygnus C Support Library*.
- a connection to the debugger. See section "Remote Debugging" in *Debugging with GDB*.

The C Run-Time Environment (`crt0`)

To link and run C or C++ programs, you need to define a small module (usually written in assembler as '`crt0.s`') that makes sure the hardware is initialized for C conventions before calling `main`.

There are some examples of '`crt0.s`' code (along with examples of system call stub code) available in the source code for your Developer's Kit. Look in the directories under:

`installdir``/progressive-96q1/src/newlib/libc/sys`

(*`installdir`* refers to your installation directory, by default '`/usr/cygnus`'.) For example, look in '`. . ./sys/h8300hms`' for Hitachi H8/300 bare boards, or in '`. . ./sys/sparclite`' for the Fujitsu SPARClite board. More examples are available under the directory:

`installdir``/progressive-96q1/src/newlib/stub`

To write your own '`crt0.s`', you need this information about your target:

- A memory map. What memory is available, and where?
- Which way does the stack grow?
- What output format do you use?

At a minimum, your '`crt0.s`' must do these things:

1. Define the symbol `start` ('`_start`' in assembler code). Execution begins at this symbol.

2. Set up the stack pointer 'sp'. It is largely up to you to choose where to store your stack within the constraints of your target's memory map. Perhaps the simplest choice is to choose a fixed-size area somewhere in the uninitialized-data section (often called 'bss'). Remember that whether you choose the low address or the high address in this area depends on the direction your stack grows.
3. Initialize all memory in the uninitialized-data ('bss') section to zero. The easiest way to do this is with the help of a linker script (see section "Command Language" in *Using LD; the GNU linker*). Use a linker script to define symbols such as 'bss_start' and 'bss_end' to record the boundaries of this section; then you can use a 'for' loop to initialize all memory between them in 'crt0.s'.
4. Call main. Nothing else will!

A more complete 'crt0.s' might also do the following:

5. Define an '_exit' subroutine (this is the C name; in your assembler code, use the label '__exit', with two leading underbars). Its precise behavior depends on the details of your system, and on your choice. Possibilities include trapping back to the boot monitor, if there is one; or to the loader, if there is no monitor; or even back to the symbol start.
6. If your target has no monitor to mediate communications with the debugger, you must set up the hardware exception handler in 'crt0.s'. See section "The GDB remote serial protocol" in *Debugging with GDB*, for details on how to use the GDB generic remote-target facilities for this purpose.
7. Perform other hardware-dependent initialization; for example, initializing an MMU or an auxiliary floating-point chip.
8. Define low-level input and output subroutines. For example, 'crt0.s' is a convenient place to define the minimal assembly-level routines described in section "System Calls" in *The Cygnus C Support Library*.