

# Porting open source (UNIX) software to OpenVMS

A product of the OpenOffice.org on OpenVMS porting group

Written by [Martin Borgman](#)

(OpenVMS Porting Guide v0.3.1)

## Table of Contents

Introduction	3
Goal	3
Intended audience	3
Open source on OpenVMS	3
OpenVMS-UNIX differences	4
GNV	8
Introduction	8
Installation	8
Quotas	10
Working with GNV	12
Hints on using the GNV tools	12
Building open source software	13
The GNU build system	14
Recommendation	14
The OpenVMS C runtime library	15
So what are these feature switches and how can I set them?	15
Recommendation	18
Macros	19
Predefined macros	19
Other macros	19
Recommendation	20
GCC wrapper	22
Porting Issues	23
Example 1: m4	35

# Introduction

## ***Goal***

This document will explain how to port open source software to OpenVMS using the features introduced in the DII COE releases of OpenVMS (7.2-6C1 & 7.2-6C2).

The document does not specifically target these versions of OpenVMS but the first public versions of OpenVMS with some of these new UNIX features. To be more specific, OpenVMS versions 7.3 7.3-1, 7.3-2 and up. OpenVMS 7.3-2 has the most of these new features.

Because more of these portability features will be added to OpenVMS in the future, we will regularly update this guide.

Because many open source software uses UNIX shell scripts to build and install itself. We will be looking on how to do the same using the UNIX commands and utilities provided by the GNV kit.

We will not look at porting open source software to older versions of OpenVMS, although some of these older versions are still supported. Porting to these older versions of OpenVMS can be done, but it is considerably more difficult.

## ***Intended audience***

Basically everybody with some C and/or C++ experience on OpenVMS or UNIX systems should be able to understand what this document will discuss.

## ***Open source on OpenVMS***

The question here is not, can we have open source software on OpenVMS, because you're already using some open source software. CDSA is an open source security framework that is now used on OpenVMS. You cannot install current Releases of OpenVMS without it. There are two CDSA add-ons, SSL and Kerberos. Both are open source. In the TCP/IP package you'll find BIND and DHCP from the Internet Software Consortium. And what about the SMTP, POP, IMAP, SSH.... These are just some packages that are part of the standard OpenVMS distribution. But there's more, there's Apache, Mozilla, Samba, MySQL, Tomcat, Perl, Python. I could continue for over an hour naming open source tools that are ported to OpenVMS.

Then there's the business rationale of open source. Why should I, or my company use open source software. This is not as simple as many people think. The fact that open source is free does not necessarily mean that it is cheaper to run for you or your company. Because of the complexity of the subject I would like to point you to a good book on the subject:

Title:               The BUSINESS and ECONOMICS of LINUX and OPEN SOURCE  
Written by:       Martin Fink  
Printed by:       Prentice Hall  
ISBN:             0-13-047677-3

## OpenVMS-UNIX differences

The best way to compare UNIX and OpenVMS is probably by way of a table.

<i>UNIX</i>	<i>VMS</i>
<p>root</p> <p>hierarchical file-system</p> <p>directory paths are separated by the / symbol</p> <p>exact paths start with /</p> <p>a relative path can start with a directory name or with .. which means go one directory back and then go to the given directory path</p> <p>the directory . is the current directory</p> <p>/ (root)</p> <p>/bin</p> <p>/etc</p> <p>/usr</p> <p>/dev</p> <p>/tmp</p> <p>/var</p>	<p>SYSTEM</p> <p>device oriented file system</p> <p>all absolute path names start with a device name or logical name followed by a dot separated path between angular brackets</p> <p>logical:[dir.subdir.subsubdir]</p> <p>originally a directory tree could only be 8 levels deep</p> <p>the root directory is always</p> <p>logical:[000000]</p> <p>a relative path is a dot separated path starting with a "." or a "-" between angular brackets</p> <p>[.dir.subdir]</p> <p>[-.dir.subdir]</p> <p>with one or more – within the angular brackets you specify you want to go one or more directory back in the directory structure</p> <p>a logical can represent a device, both actual devices, points in the file system we want to represent as devices (concealed), paths, path lists or other values or value lists</p> <p>SYS\$SYSDEVICE: is an example of a device logical. It is the device holding the system software</p> <p>SYS\$COMMON: is an example of a path that can be used as a device (you can do DIR SYS\$COMMON:[000000])</p> <p>SYS\$SYSROOT: is an example of a path list</p> <p>SYS\$ANNOUNCE: is an example of a simple value</p> <p>with ... at the end of a path you specify all subdirectories.</p> <p>You can use % as single character wild card character and * for multiple character wild card characters both in filenames and in file paths.</p> <p>XX%%YY.*</p> <p>*SMTP*.COM</p>
<p>Numerous filesystems: ufs (the original UNIX filesystem), ext, ext2, ext3 (Linux), adfs (True64), efs(IRIX), hfs, s5 (HP-UX), jfs (AIX) and let's not forget nfs.</p>	<p>The most important ones for this document are ODS-2 and ODS-5. ODS-5 is rather new and is a requirement for UNIX portability.</p>

<i>UNIX</i>	<i>VMS</i>
Symbolic links	<p>Hardlinks or Softlinks</p> <p>Hardlinks were introduced in 7.3-1 and only work when enabled on the disk.</p> <p>The advantage over softlinks is that a reference count is maintained, to make sure that the file doesn't get deleted until the last link is deleted.</p>
pwd	SHOW DEFAULT
df	SHOW DEV D
cd	SET DEFAULT
/bin \$HOME or ~ \$PATH	<p>SYS\$SYSTEM:</p> <p>SYS\$LOGIN:</p> <p>DCL\$PATH:</p>
environment variables	<p>symbols &amp; logicals</p> <p>symbols are quite similar to UNIX environment variables</p> <p>logicals can also contain values but they are not necessarily limited to the scope of your process or job. Logicals are organized in tables that can be seen cluster wide, system wide, group wide, process wide or job specific. You can even create your own logical name tables.</p>
case sensitive	<p>OpenVMS is case insensitive by default. Since version 7.3 you can make your process behave case preservative on ODS-5 disks by enabling extended filename parsing. Since version 7.3-1 you can also make your process behave case sensitive on ODS-5 disks.</p> <pre>SET PROCESS /PARSE_STYLE=(TRADITIONAL/EXTENDED)  SET PROCESS /CASE_LOOKUP=(BLIND/SENSITIVE)</pre>
character devices, block devices	<p>mailbox (qio/qiow) RMS</p> <p>...</p>

<i>UNIX</i>	<i>VMS</i>
/dev/..., makedev, mount, fstab.	<p>VMS recognizes all new devices at boot time and automatically loads the necessary drivers.</p> <p>It is possible to load drivers and add new devices to a running system. OpenVMS 7.3-2 even allows you to enlarge disk volumes dynamically.</p> <p>To use a new disk you can use the following command to initialize it:</p> <pre>\$ INITIALIZE &lt;device&gt; &lt;label&gt; - /NOHIGHWATER /STRUCTURE=5 - /VOLUME_CHARACTERISTICS= - (HARDLINKS,ACCESS_DATES)</pre> <p>Mount the disk with the following command:</p> <pre>\$ MOUNT &lt;device&gt; &lt;label&gt; /SYSTEM - /NOASSIST</pre> <p>To mount a disk every time when the system boots add the above command to SYSS\$MANAGER:SYSTARTUP_VMS.COM.</p>
sh, csh, tcsh, bash, ksh, zsh, ...	DCL
vi	EDIT (/TPU is the default)
/etc/profile	SYSS\$MANAGER:SYLOGIN.COM
~/.profile	SYSS\$LOGIN:LOGIN.COM
chmod +x to make a file executable	No equivalent, see below
.<filename> will execute an executable file (script or program) when not in your PATH environment	<p>@ to execute a DCL script</p> <p>RUN to execute a program not somehow known to your runtime environment.</p> <p>&lt;command&gt; to execute a program known to your runtime environment.</p> <p>There are various ways to extend your runtime environment.</p> <ol style="list-style-type: none"> <li>1. define a symbol that points to the program to execute i.e. UNZIP :=  SYSS\$LOGIN:[TOOLS]UNZIP.EXE. UNZIP is now a command known to your environment.</li> <li>2. add the directory containing the executable to your DCL\$PATH i.e. DEFINE DCL\$PATH  SYSS\$LOGIN:[TOOLS], 'F\$TRNLNM("DCL\$PATH")'. Now every executable program in the directory SYSS\$LOGIN:[TOOLS] is available as a command in your runtime environment.</li> <li>3. Add a command definition to to your environment i.e. SET COMMAND.</li> </ol>

This is but a partial list. It is probably best to point you to some good books on the subject:

Title: Linux and OpenVMS Interoperability

Written by: John Wisniewski

Printed by: Digital Press

ISBN: 1-55558-267-2

Title: UNIX for VMS Users  
Written by: Philip E. Bourne  
Printed by: Digital Press  
ISBN: 1-55558-034-3

# GNV

## Introduction

GNV stands for GNU is not VMS. It is a set of open source commands and utilities that are ported to OpenVMS. The GNV kit contains a port of the bash shell, the gnu diff utilities, the gnu file utilities, the gnu find utilities, gawk, grep, gzip, less, gnu make, man, sed, gnu shell utilities, gnu text utilities, (un)zip, a vi wrapper for tpu, vms(un)tar and wrappers for ar, cc, gcc and cpp. This list is not complete and more UNIX utilities become available with every new version of the GNV kit.

GNV is an open source project with a web page <<http://gnv.sourceforge.net>> and several mailing lists.

If you are planning to add tools to GNV, join the gnv developer list.

## Installation

First download the latest kit from the OpenVMS open source page at <<http://www.openvms.compaq.com/opensource>> or from the GNV project page at <<http://gnv.sourceforge.net>>.

Print the installation instructions and use them to install the kit. Please read it carefully because it contains some important information.

The requirements for installing and using GNV are:

- must install on an ODS-5 disk
- must use on an ODS-5 disk

To check the installation disk use the following command:

```
$ SHOW DEVICE DSA0: /FULL

Disk DSA0:, device type COMPAQ BB00911CA0, is online, mounted, file-oriented
device, shareable, available to cluster, error logging is enabled, device
supports bitmaps (no bitmaps active).

Error count                0      Operations completed          566976
Owner process              ""      Owner UIC                  [SYSTEM]
Owner process ID          00000000  Dev Prot                   S:RWPL,O:RWPL,G:R,W
Reference count           664      Default buffer size        512
Total blocks              17773524 Sectors per track          168
Total cylinders           5290     Tracks per cylinder         20
Logical Volume Size       17773524 Expansion Size Limit        17793024

Volume label              "ALPHASYS" Relative volume number       0
Cluster size              3        Transaction count           439
Free blocks              14097399  Maximum files allowed       2221690
Extend quantity          5        Mount count                  1
Mount status              System   Cache name                   "_DSA0:XQPCACHE"
Extent cache size         64      Maximum blocks in extent cache 1409739
File ID cache size        64      Blocks in extent cache       113424
Quota cache size          0        Maximum buffers in FCP cache  1720
Volume owner UIC          [1,1]    Vol Prot                    S:RWCD,O:RWCD,G:RWCD,W:RWCD

Volume Status:  ODS-5, subject to mount verification, protected subsystems
enabled, write-through caching enabled, hard links enabled.
```

And this is not all the information but the important bit is at the bottom of the example text. Right after the Volume Status: header it says ODS-5 in this case. So this disk should be OK for use. Also note that hard links are enabled. For UNIX portability is is generally a good idea to have this enabled. It is also a good idea to enable access dates on a volume because this is a POSIX requirement.



To make a disk ODS-5 compatible use the following command:

```
$ SET VOLUME <device>: /STRUCTURE=5
```

To enable hard links and access dates use the following command:

```
$ SET VOLUME <device>: /VOLUME_CHARACTERISTICS=(HARDLINKS, ACCESS_DATES)
```

Be aware that enabling hard links can take a considerable amount of time!

Another thing I would like to note here is high-water marking. This is another typical OpenVMS security feature that is on by default. High-water marking guarantees that a user cannot read data that was not written by the user, by destroying all data in the disk block after the EOF marker. There is a small performance penalty in doing this. Although the penalty is only marginal these days, many people will advise you to turn it off. You can do so by giving the following command:

```
$ SET VOLUME <device>: /NOHIGHWATER_MARKING
```

Disabling high-water marking is not mandatory for the installation of GNV or the UNIX portability.

To do a default install of the current GNV software, follow these steps:

1. Log in to the SYSTEM account (at the login prompt, enter user name SYSTEM and the appropriate password), or an account with equivalent privileges.
2. At the DCL prompt (\$), go to the directory where you saved the executable you downloaded from the Internet and extract the PCSI kit by running the executable:

```
$ RUN dec-axpvms-gnv-v0105-006-1.pcsi-dcx_axpexe
```

3. Type the following command, as shown:

```
$ PRODUCT INSTALL GNV
```

4. When you enter the PRODUCT INSTALL command, the system responds with a display similar to the following:

```
The following product has been selected:
DEC AXPVMS GNV V1.5                      Layered Product
Do you want to continue? [YES]
```

Continue the procedure by pressing the ENTER key for the default answer (YES). The procedure might take several minutes and numerous messages might be displayed on the screen. In response to each prompt displayed by the system, choose the default answer by pressing the ENTER key.

5. To use GNV we need to do some system wide setup. To do this we start the following DCL command procedure:

```
$ @SYS$STARTUP:GNV$STARTUP.COM
```

To do this setup every time we boot our system we need to add the above command to the system startup procedure SYS\$MANAGER:SYSTARTUP\_VMS.COM . So fire up the editor:

```
$ EDIT SYS$MANAGER:SYSTARTUP_VMS.COM
```

Go to the bottom of the file (<Do>bottom) and enter the following lines before the \$ EXIT:

```
$!
$! GNV
$!
$ file := SYS$STARTUP:GNV$STARTUP.COM
$ IF F$SEARCH(file) .NES. "" THEN @'file'
$!
```

And exit the editor with <Ctrl>Z.

6. There's also some user setup that we'll need to do. For our current login session do the following:

```
$ IF F$TRNLNM("GNU","LNM$SYSTEM_TABLE") .NES. "" THEN @GNU:[LIB]GNV_SETUP.COM
```

And if we want this user setup to happen when you log in, we must add the above line to our SYS\$LOGIN:LOGIN.COM. This counts for every user on your system who wants to use the GNV tools. If we want this to happen for every user on our system, we could add the line to the system wide login procedure SYS\$MANAGER:SYLOGIN.COM (\$ EDIT 'F\$TRNLNM("SYS\$SYLOGIN")'.COM). Just remember to add the line before the EXIT statement (login scrips usually do thing depending on the current mode login (F\$MODE), note that the gnv setup procedure must be made to work in interactive mode if you plan to use it on the command line).

And of course, no reboot is needed.

## Quotas

Managing quotas is another OpenVMS stronghold. It allows the system to manage the system resources and maintain an acceptable performance for all processes on the system.

The following recommendation comes from the June 2003 issue of "Optimizing Java Technology Software Performance on HP OpenVMS". You can find this document at the HP OpenVMS e-Business Technology page <http://h71000.www7.hp.com/ebusiness/technology.html>.

Please note that the suggested quota's are a minimum requirement for UNIX compatibility.

## Account quotas

Log is as user SYSTEM and go to the SYS\$SYSTEM directory:

```
$ SET DEFAULT SYS$SYSTEM:
```

Start the account management program:

```
$ MC AUTHORIZE
```

Show your account settings (in the example below the DEFAULT account is shown, this is a special account used for account creation):

```
UAF> sho default

Username: DEFAULT                                Owner:
Account:                                         UIC:      [200,200] ([DEFAULT])
CLI:      DCL                                    Tables: DCLTABLES
Default:   [USER]
LGICMD:
Flags: DisUser
Primary days: Mon Tue Wed Thu Fri
Secondary days:                               Sat Sun
No access restrictions
Expiration:      (none)      Pwdminimum: 6      Login Fails:      0
Pwdlifetime:     90 00:00    Pwdchange:      (pre-expired)
Last Login:      (none) (interactive),          (none) (non-interactive)
Maxjobs:         0  Fillm:    100  Byt1m:        64000
Maxacctjobs:     0  Shrfillm: 0  Pbyt1m:         0
Maxdetach:       0  BIO1m:    150  JTquota:       4096
Prclm:           8  DIO1m:    150  WSdef:         2000
Prio:            4  AST1m:    250  WSquo:         4000
Queprio:         4  TQElm:     10  WSextent:      16384
CPU:             (none) Enqlm:   2000 Pgflquo:      50000
Authorized Privileges:
  NETMBX      TMPMBX
```

```
Default Privileges:
NETMBX      TMPMBX
UAF>
```

We are going to change the following minimum account quotas:

<i><b>Parameter Name</b></i>	<i><b>Value</b></i>
FILLM	4096
WSDEF	2048
WSQUOTA	4096
WSEXTENT	16384
WSMAX	16384
PGFLQUO	2097152
BYTLM	400000
BIOLM	150
DIOLM	150
TQELM	100

To change the values of the parameters do the following:

```
UAF> MODIFY <account> /<parameter>=<value>
```

You can change multiple parameters in the same command just add another /<parameter>=<value> for each value you need to change. To change the WSDEF, WSQUO, WSEXTENT and WSMAX for the DEFAULT account you would enter the following command:

```
UAF> MODIFY DEFAULT /WSDEF=2048 /WSQUO=4096 /WSEXTENT=16384 /WSMAX=16384
```

You exit the AUTHORIZE utility by typing EXIT, or <Ctrl>Z (It may be possible that the terminal emulator you are using correctly maps the EXIT function to the F10 function key) .

As said before, the DEFAULT account is used as a template for the creation of new accounts. If you want to raise the default quotas for all new accounts you are going to create on your system, it may be a good idea to modify the DEFAULT account.

Changing the default account will not change any setting for existing accounts. Check and modify existing accounts as needed

To use the changed account settings, you must log out and back in again.

## System parameters

OpenVMS system parameters are very much like UNIX kernel parameters.

CHANNELCNT

Page file size

PQL\_

other??

SYSMAN or AUTOGEN?

## Working with GNV

When you have installed GNV correctly you should be able to use UNIX commands like `ls` right from DCL. However if you have LSE installed the `ls` command will start the LSE editor instead.

When you type `bash` at the DCL prompt. The bash shell is started and from that point on everything works as if you were using a UNIX box. There are however some minor but important differences. You may know that UNIX systems use a hierarchical file-system and OpenVMS does not. However some of the important hierarchy is mimicked under OpenVMS. By default the `/` directory points to the OpenVMS GNU:[000000] directory. The `/bin` directory points to the OpenVMS GNU:[bin] directory. Etc. In short the `/` directory is not the root of a disk!

You can access other locations by typing `/device-name or concealed device logical/directory`. I.E. `DSA50:[kits.gnu]` becomes `/dsa50/kits/gnu`.

When in bash you can enter most DCL commands, unless there is a name conflict like the `ls` LSE problem. In that case you can enter the `bash dcl` command and enter the DCL command you wanted to use behind it. I.E. the DCL `LS Readme` command becomes `dcl "ls Readme"` in bash. You can also turn this behavior off by doing the following in BASH:

```
bash$ export GNV_DISABLE_DCL_FALLBACK=1
```

## Hints on using the GNV tools

- always work on ODS-5 disks  
`$ SET VOLUME <device>: /STRUCTURE=5`
- always enable ODS-5 extended filename parsing  
`$ SET PROCESS /PARSE_STYLE=EXTENDED`  
(you can put this line in your `LOGIN.COM`)
- because of the way bash handles pipes at the moment we need to do the following  
`$ DEFINE/JOB DECC$PIPE_BUFFER_SIZE=65000`  
(you can put this line in your `LOGIN.COM`)
- the best way to handle UNIX symbolic links is to use hard links on the disk you are working from  
`$ SET VOLUME <device>: /VOLUME_CHARACTERISTICS=(HARDLINKS)`  
Please note that this can take a considerable amount of time.
- it may be necessary to support POSIX style access dates  
`$ SET VOLUME <device>: /VOLUME_CHARACTERISTICS=(ACCESS_DATES)`
- when running configure scripts, it may be a good idea to disable DCL fall back  
`bash$ export GNV_DISABLE_DCL_FALLBACK=1`  
(you can put this in your `.profile`)

If you want to know more about using UNIX or bash, there is plenty of information available on the Internet. If you like books better, take a look at the offerings from O'Reilly <<http://www.oreilly.com/>>.

I suggest you do take some time to learn the UNIX environment before continuing.

# Building open source software

Most of you probably know that open source means that the software is usually distributed in source form. To use the software on your system you need to build it yourself.

To make this building much easier the GNU community developed the GNU build tools. By the way, these tools not only make your life much easier, they also make the life of the open source developers much easier.

Now let's look at how you would build some open source software on a UNIX or Linux box.

1. Download the source distribution from the Internet. Most of the time this will be a .tar.gz file but you'll also find .tgz, .tar.Z .tar.bz2 or .zip files.
2. Create a directory  
% mkdir <name>
3. Go to the directory you just created  
% cd <name>
4. Unpack the file (some options may not work on OpenVMS yet)
  - .tar.gz      % zcat <filename> | tar xvf -  
              or      % tar xvzf <filename>  
              or      % gunzip <filename>  
                      % tar xvf <filename - .gz>
  - .tgz            see .tar.gz
  - .tar.Z          see .tar.gz  
              or      % uncompress <filename>  
                      % tar xvf <filename - .Z>
  - .tar.bz2        % bz2cat <filename> | tar xvf -  
              or      % bunzip2 <filename>  
                      % tar xvf <filename - .bz2>
  - .zip            % unzip <filename>
5. look for the configure script it may be in the directory you're in, but it may also be in the directory that was created by unpacking the distribution file. I.e. If the file you downloaded from the Internet was called tar-1.2.4.tar.gz it is possible that you now have a directory named tar-1.2.4 in your current directory. Enter that directory to look for a configure file. If you don't find one look in the files with uppercase names like README or INSTALL.
6. When you found the configure script, run it  
% ./configure
7. When all go's well you can build the executable(s)  
% make
8. And install the package (usually you must first become the root user before you can install something)  
% make install

And that all folks.

Well, that was all when everything works the way it's supposed to. And even on a UNIX box things can go wrong. To analyze what went wrong do the following:

- read the files with uppercase names and find out if you meet the prerequisites, find out if you need to do something special for your UNIX version (after all UNIX != UNIX)
- try ./configure --help this may give you an idea about extra parameters you may have to specify with the ./configure command
- check the files that were generated by the configure script: config.cache, config.log, config.status, config.h and Makefile(s)

If this didn't help you may need to make your hands dirty.

## ***The GNU build system***

Didn't I say before that the GNU build system would make your life much easier? Well it does. But let me first explain what may go wrong in the last example. The configure script and the Makefile.in file(s) may have been created before your UNIX system or system version came into existence, so it doesn't know about your systems specifics.

Thinking about OpenVMS for a moment, probably none of the configure scripts and make files out there, know anything about our GNV environment on OpenVMS

But let's get back to the problem at hand.

To make the configure script and the Makefile.in file(s) aware of our systems specifics we will need to add some steps to our build example.

The following steps come between steps 5 and 6:

- a. % `aclocal`
- b. % `autoconf`
- c. % `automake -a`

To make this work, your system should have these packages and GNU make, m4, texinfo and GNU tar readily installed. For most open source UNIX clones like Linux, FreeBSD, NetBSD and MacOS X, this is true. But for many commercial UNIX systems this isn't the case. As for OpenVMS, only GNU make is available at the moment.

The best thing for most commercial UNIX systems is to check out if the manufacturer of your UNIX system made the missing tools available for download. If not check out the latest versions of the missing tools from the GNU web site. If nothing worked out, you are in the same position as us with our GNV environment on OpenVMS.

At this point I think that it should be clear to us that we need all these tools a.s.a.p.

If you want to know more about the gnu build system, take a look at the following web site(s):

GNU Autoconf, Automake and Libtool: <http://sources.redhat.com/autobook/>

## ***Recommendation***

Because of a bug in GNV version 1.5-6 bash, I recommend to get the latest GNV sources from the CVS repository and build the GNV kit yourself. See the build instructions on <http://www.4ovms.dyndns.org>

Don't forget to raise the pipe buffer size.

```
bash$ dcl "define/job DECC\${PIPE_BUFFER_SIZE} 65000"
```

And don't forget to disable DCL fallback.

```
bash$ export GNV_DISABLE_DCL_FALLBACK=1
```

Always check the configure script for conftest.dir. Change the name of this file to something like conftest.ddd

```
bash$ mv configure configure.org
bash$ sed 's/conftest\.dir/conftest\.ddd/g' configure.org > configure
```

Use the configure option `--build=<Processor>-<OS maker>-<OS name>`. For OpenVMS Alpha the configure option is `--build=alpha-hp-vms`.

Because the GNV bash version is somewhat outdated, some shell scripts do not work. One of those scripts is called depcomp. This script is used to determine dependency's during compilation. To get rid of this dependency checking, you can add the `--disable-dependency-tracking` option to the configure command.

## The OpenVMS C runtime library

The C Run Time Library or CRTL is an OpenVMS shared library containing most of the “standard” C functions. I quoted the word standard because a) there are a lot of standards and I didn't say which one. b) every standard is bit of a moving target.

Since the DII COE initiative, HP intends to make the CRTL compliant to the latest X-Open standard. Because of the architectural differences between OpenVMS and UNIX this is not an easy task, especially if you also want to maintain backwards compatibility.

The OpenVMS CRTL developers decided to add these new features gradually with every new OpenVMS and CRTL release.

HP also decided that these new features will become available on both the Alpha and the new Itanium architecture. Some features may also become available on the VAX, but because ODS-5 extended filename support is not available on the VAX, the functionality that is essential to open source porting will probably never become available on the VAX.

The new CRTL features can be categorized in two groups:

1. existing functions with new and more standard functionality
2. completely new functions to make the CRTL more standard (X-Open v.6) compliant.

For the functions in group one the classic behavior is standard and to use the new functionality you need to set feature switches (more on those later).

The functions in group two can be divided in two more groups

1. functions that should work the same in both the classic OpenVMS environment as in the new UNIX like environment
2. functions that should when not work the same in both environments

For this last group, think about functions that use or return a path. For this group the behavior of these functions is determined by the previously mentioned feature switches.

### ***So what are these feature switches and how can I set them?***

I will not explain what feature switches there are and what they do exactly because you can read all about them in the CRTL help, the CRTL release notes and the CRTL Reference manual. But I will show you some of the interesting new ones in OpenVMS 7.3-2. And I will show you how you can change their values.

So let's first get to the most interesting new feature switch, especially when you're new to porting. It is DECC\$UNIX\_LEVEL. The text below is from the CRTL reference manual (CRTL 1-38):

With the DECC\$UNIX\_LEVEL logical name, you can manage multiple C RTL feature logical names at once. By setting a value for DECC\$UNIX\_LEVEL from 1 to 100, you determine the default value for groups of feature logical names. The value you set has a cumulative effect: the higher the value, the more groups that are affected. Setting a value of 20, for example, enables all the feature logicals associated with a DECC\$UNIX\_LEVEL of 20, 10 and 1.

The principal logical names affecting UNIX like behavior are grouped as follows:

- |    |  |
|----|--|
| 1  | General corrections                            |
| 10 | Enhancements                                   |
| 20 | UNIX style file names                          |
| 30 | UNIX style file attributes                     |
| 90 | Full UNIX behavior - No concessions to OpenVMS |

Level 30 is appropriate for UNIX like programs such as BASH and GNV.

The DECC\$UNIX\_LEVEL values and associated groups of affected feature logical names are:

```

General Corrections          (DECC$UNIX_LEVEL == 1)
DECC$FIXED_LENGTH_SEEK_TO_EOF      1
DECC$POSIX_SEEK_STREAM_FILE      1
DECC$SELECT_IGNORES_INVALID_FD    1
DECC$STRTOL_ERANGE                1
DECC$VALIDATE_SIGNAL_IN_KILL      1

General Enhancements        (DECC$UNIX_LEVEL == 10)
DECC$ARGV_PARSE_STYLE            1
DECC$EFS_CASE_PRESERVE           1
DECC$STDIO_CTX_EOL               1
DECC$PIPE_BUFFER_SIZE            4096
DECC$USE_RAB64                   1

UNIX style file names       (DECC$UNIX_LEVEL == 20)
DECC$DISABLE_TO_VMS_LOGNAME_TRANSLATION 1
DECC$EFS_CHARSET                 1
DECC$FILENAME_UNIX_NO_VERSION    1
DECC$FILENAME_UNIX_REPORT        1
DECC$READDIR_DROPDOTNOTYPE       1
DECC$RENAME_NO_INHERIT           1
DECC$GLOB_UNIX_STYLE             1

UNIX like file attributes    (DECC$UNIX_LEVEL == 30)
DECC$EFS_FILE_TIMESTAMPS         1
DECC$EXEC_FILEATTR_INHERITANCE  1
DECC$FILE_OWNER_UNIX             1
DECC$FILE_PERMISSION_UNIX        1
DECC$FILE_SHARING                1

UNIX compliant behavior      (DECC$UNIX_LEVEL == 90)
DECC$FILENAME_UNIX_ONLY          1
DECC$POSIX_STYLE_UID             1
DECC$USE_JPI$_CREATOR            1
DECC$DETACHED_CHILD_PROCESS      1

```

#### Notes:

- Defining a logical name for an individual feature logical supersedes the default value established by DECC\$UNIX\_LEVEL for that feature.
- Future revisions of the C RTL may add new feature logicals to a given DECC\$UNIX\_LEVEL. For applications that specify that UNIX level, the effect is to enable those features by default.

Please note that not all the available feature switches are listed in the quote. The ones not listed are not part of any UNIX level! Read the CRTL help, release notes and documentation for a full list of supported feature switches and their intended use.

You may already have noticed one way of setting these feature switches. You can define a logical with the same name and appropriate value. And because the feature switches alter the behavior at runtime, you don't have to recompile while playing with the switches. But when you are finished determining the appropriate values for the switches you may want to set their values from within your program. You can of course do this by setting the logical from within your program, but there is a much better way.

The CRTL functions `decc$feature_get_index`, `decc$feature_get_name`, `decc$feature_get_value`, and `decc$feature_set_value` are specially designed for this task.

Below you'll see a small example (CRTL REF-82): program.

```

static int set_feature_default(char *name, int value)
{
    int index = decc$feature_get_index(name);
    if (index == -1 || decc$feature_set_value(index, 0, value) == -1)
    {
        perror(name);
        return -1;
    }
}

```



```

    }
    return 0;
}

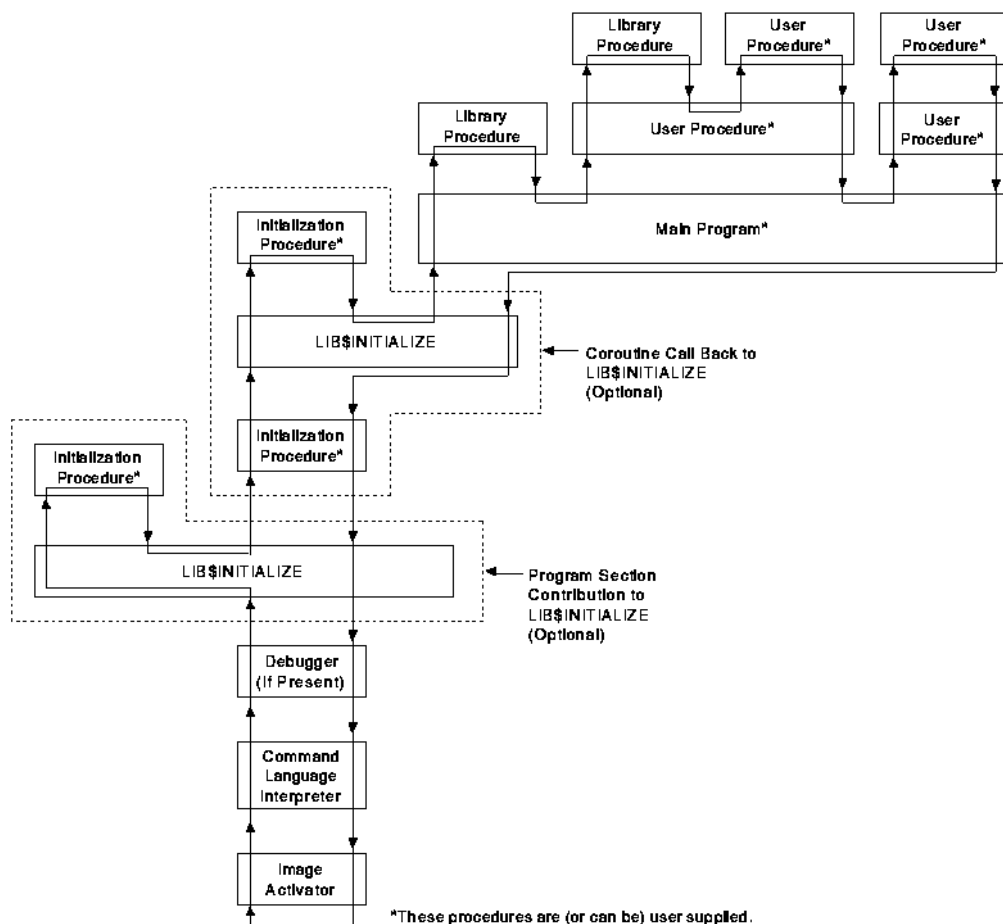
static void my_init( void)
{
    set_feature_default("DECC$POSIX_SEEK_STREAM_FILE" , TRUE);
    set_feature_default("DECC$ARGV_CASE_PARSE_STYLE" , TRUE);
    set_feature_default("DECC$EFS_CASE_PRESERVE" , TRUE);
    set_feature_default("DECC$FILE_SHARING" , TRUE);
}

```

It is not too difficult to add code like this to a program, but there's a catch.

Feature switches like DECC\$ARGV\_CASE\_PARSE\_STYLE need to be set before the arguments are being parsed and this happens somewhere between image activation and the call to the programs main function.

The process of what happens before your program starts is documented in the OpenVMS Programming Concepts Manual. In chapter 18 you'll find a complete flow of what happens and when (see picture below).



ZK-1977-GE

The important thing to note here is the LIB\$INITIALIZE function. What we are going to do is create a source file that defines a LIB\$INITIALIZE function that points to a function that sets the feature switches to their appropriate values.

Now let's go back to the example I used above. An enhanced version of the example on page CTRL REF-83 is presented below.

```
#pragma extern_model save
#pragma extern_model strict_refdef "LIB$INITIALIZE" nowrt, long
#if __INITIAL_POINTER_SIZE
#   pragma __pointer_size __save
#   pragma __pointer_size 32
#else
#   pragma __required_pointer_size __save
#   pragma __required_pointer_size 32
#endif
/* Set our contribution to the LIB$INITIALIZE array */
void (* const iniarray[])() = {my_init, } ;
#if __INITIAL_POINTER_SIZE
#   pragma __pointer_size __restore
#else
#   pragma __required_pointer_size __restore
#endif
#pragma extern_model restore

/*
** Force a reference to LIB$INITIALIZE to ensure it
** exists in the image.
*/
int LIB$INITIALIZE();
globaldef int (*lib_init_ref)() = LIB$INITIALIZE;
```

Put both parts of the example in one source file. Compile and link with your program to make it work. The beauty of it is that you don't have to change the program you are porting to set the feature switches!

If you want to see a more complete example, take a look at GNU:[src.GNV.CRTLSUP.SRC]VMS\_CRTL\_INIT.C file. This file is used by some of the GNV tools.

## **Recommendation**

A good starting point is UNIX level 30, but don't use the DECC\$UNIX\_LEVEL feature switch. Set the individual feature switches that make up UNIX level 30 and increase DECC\$PIPE\_BUFFER\_SIZE to at least 8192. The reasons for not using the DECC\$UNIX\_LEVEL feature switch are compatibility with OpenVMS 7.3-1 and the simple fact that you may not need all the feature switches that make up level 30 or you want to use different values (see the recommendation for DECC\$PIPE\_BUFFER\_SIZE).

# Macros

## Predefined macros

The list below is a list of predefined macros on OpenVMS 7.3-2 Alpha using the Standard DECC C compiler version 6.5.

These macros are in effect at the start of the compilation.

```
-----

__G_FLOAT=1 __DECC=1 vms=1 VMS=1 __32BITS=1 __PRAGMA_ENVIRONMENT=1
__CRTL_VER=70320000 __vms_version="V7.3-2 " CC$gfloat=1 __X_FLOAT=1
vms_version="V7.3-2 " __DATE__="Feb 7 2004" __STDC_VERSION__=199901L
__DECC_MODE_RELAXED=1 __DECC_VER=60590001 __VMS=1 __ALPHA=1
VMS_VERSION="V7.3-2 " __IEEE_FLOAT=0 __VMS_VERSION="V7.3-2 "
__STDC_HOSTED__=1 __TIME__="13:54:21" __Alpha_AXP=1 __VMS_VER=70320022
__BIASED_FLT_ROUNDS=2 __INITIAL_POINTER_SIZE=0 __STDC__=1
__LANGUAGE_C__=1 __vms=1 __alpha=1 __D_FLOAT=0
```

The following macros are important for porting:

__VMS, __vms	Are the ones we can use for general OpenVMS specific stuff
__ALPHA, __Alpha_AXP, __alpha	Can be used for Alpha specific stuff
__VAXC, __VAX11C, __vaxc, __vax11c	Can be used for VAX specific stuff
__IA64, __ia64	Can be used for Itanium (ia64) specific stuff
__VMS_VER	Can be used for OpenVMS version specific stuff (please use __CRTL_VER instead)
__DECC_VER	Can be used for compiler version specific stuff
__CRTL_VER	Can be used for CRTL version specific stuff

Please note that I only included the preferred format of the macros, most of them also exist without the two dashes in front.

## Other macros

There are also numerous other macros that change the behavior of one or more CRTL functions.

**\_\_XOPEN\_SOURCE\_EXTENDED** Makes visible XPG4-extended features, including traditional UNIX based interfaces not previously adopted by X/Open.

Standard Selected: XPG4 V2

Other Standards Implied: XPG4, ISO POSIX-2, ISO POSIX-1, ANSI C

<code>_XOPEN_SOURCE</code>	Makes visible XPG4 standard symbols and causes <code>_POSIX_C_SOURCE</code> to be set to 2 if it is not already defined with a value greater than 2. Standard Selected: XPG4 Other Standards Implied: XPG4, ISO POSIX-2, ISO POSIX-1, ANSI C
<code>_POSIX_C_SOURCE==199506</code>	Header files defined by ANSI C make visible those symbols required by IEEE 1003.1c-1995. Standard Selected: IEEE 1003.1c-1995 Other Standards Implied: ISO POSIX-2, ISO POSIX-1, ANSI C
<code>_POSIX_C_SOURCE==2</code>	Header files defined by ANSI C make visible those symbols required by ISO POSIX-2 plus those required by ISO POSIX-1. Standard Selected: ISO POSIX-2 Other Standards Implied: ISO POSIX-1, ANSI C
<code>_POSIX_C_SOURCE==1</code>	Header files defined by ANSI C make visible those symbols required by ISO POSIX-1. Standard Selected: ISO POSIX-1 Other Standards Implied: ANSI C
<code>_STDC_VERSION__==199409_</code>	Makes ISO C Amendment 1 symbols visible. Standard Selected: ISO C Amendment 1. Other Standards Implied: ANSI C
<code>_ANSI_C_SOURCE</code>	Makes ANSI C standard symbols visible. Standard Selected: ANSI C. Other Standards Implied: None.
<code>_POSIX_EXIT</code>	To enable the ISO POSIX-1 compatible exit function.
<code>_BSD44_CURSES</code>	This macro selects the Curses package from the 4.4BSD Berkeley Software Distribution.
<code>_VMS_CURSES</code>	This macro selects a Curses package based on the VAX C compiler. This is the default Curses package.
<code>_SOCKADDR_LEN</code>	This macro is used to select 4.4BSD-compatible and XPG4 V2-compatible socket interfaces. These interfaces require support in your underlying TCP/IP software. Contact your TCP/IP vendor to inquire if the version of TCP/IP software you run supports 4.4BSD sockets. (HP TCP/IP supports this feature, as far as I can tell, Prossess Softwares Multinet and TCPware don't)
<code>_LARGEFILE</code>	The C RTL provides support for compiling applications to use file sizes and offsets that are two gigabytes (GB) and larger. This is accomplished by allowing file offsets of 64-bit integers.
<code>__USE_LONG_GID_T</code>	To compile an application for 32-bit UID/GID support.
<code>_DECC_SHORT_GID_T</code>	To compile an application for 16-bit UID/GID support

Please read the CRTL help and/or CRTL Reference Manual for more information.

## Recommendation

Please don't use general macros like `VMS`, `vms`, `__VMS` or `_vms`. Use `__CRTL_VER` instead. The C Runtime Library is becoming more "standard" with every new version of the CRTL. Fixes may no longer be necessary in newer versions of the CRTL. In some cases program functionality has been crippled with `#ifndef VMS` in a way that is no longer necessary.

Using `__CRTL_VER` will allow us to control more precisely what needs to be changed for the many different versions of the CRTL.

When the macro `__CRTL_VER` doesn't exist, define it with the value of `__VMS_VER`.

```
# if defined(__VMS_VER) && !defined(__CRTL_VER)
```

```
#   define __CRTL_VER __VMS_VER
# endif
```

When compiling open source programs in DCL, add the /DEFINE=\_POSIX\_EXIT to your CC command! This is the default when compiling under bash.

## **GCC wrapper**

The GNV kit does not come with gcc but with a wrapper that uses HP C and C++ compilers to do its compilation work and it uses the standard OpenVMS Linker as a substitute for ld. By the way there is also a ar wrapper that uses the standard OpenVMS Librarian.

The GCC wrapper does quite a good job, but it isn't perfect. Also note that the HP compilers are very strict. By this I mean that they complain immediately when something may not be entirely correct. So we'll see a lot of warnings when compiling open source software.

You can see all the options the gcc wrapper supports by using the -h or -help option.

I would like to note the following on the behavior of the GCC wrapper, the wrapper adds the /DEFINE=\_POSIX\_EXIT to the compile statement by default.

The -Wc and -Wl gcc option may come in handy if you want to add a HP CC, CPP or LINK option to the gcc command line, that does not seem to be implemented.

One gcc option you sometimes need is -names\_as\_is\_short. This will make the compiler case sensitive and it allows function names with more than 32 characters.

You may get into trouble with the gcc -g option in configure scripts. Many configure scripts use this option by default. This option is used to compile and/or link with debug information included in the program. The problem is that on UNIX systems a program compiled and linked with debug information doesn't start the debugger when executed. This is quite different from what we OpenVMS people are used to. If you see the debugger prompt pop up when running a configure script, either remove the -g option from the configure script or start configure with CFLAGS="".

## Porting Issues

Quite a few CRTL functions still don't quite work as their UNIX counterparts and some functions are still missing. To give you some examples:

The stat structure returned by the stat, fstat and lstat functions is not the same as the one used on most UNIX systems. On OpenVMS the fields st\_dev and st\_rdev are pointers to a string containing the physical device name. On UNIX systems these fields contain the device number.

To continue with the stat structure the field st\_ino is defined on OpenVMS as tree unsigned short values and on UNIX systems this is usually an unsigned integer (32bit) You'll probably understand that these incompatibilities can cause some interesting porting issues.

Another example I would like to note is vfork. This function does not behave exactly as its UNIX counterpart (read the section on vfork in the CRTL reference manual). If a program uses vfork it will probably check the existence of vfork during configure and this is also a situation where configure will hang. The solution to the hang is quite simple. Log in using a different session and look for the child process of your previous session. Kill it and the configure script continues.

And while we're at it, OpenVMS unfortunately has no fork function yet, but there are many known workarounds available is only we know where.

This is not a definitive guide on how to port open source to OpenVMS. Things are changing quite rapidly. We should continue to keep this document up to date with the current state of OpenVMS and the GNV kit.

Below is a list, found on comp.os.vms, of known areas where compatibility problems occur when porting from UNIX to OpenVMS.

Legend:

issue#: sequential number (1...31 from cov list, rest locally generated)  
sev/prio: severity to our port (VMS encoded..)/our priority rating (VMS encoded..)  
The severity and priority are encoded from the OpenOffice porting group viewpoint.  
(default = I/O)  
status: VMS encoded activity from the OpenOffice group viewpoint.  
CUR: actively being worked on.  
COM: no local activity due to lack of local resources.  
LEF: waiting for something under local control (f.i. Media).  
CEF: waiting for something under external control (f.i. patch from VMS-engineering).  
(default = LEF)  
description: Short description of the issue.  
explanation, Explanation of the issue and known workarounds, fixes and viewpoints.  
status:

We added some comments to some of the items (see (OO)).

<i>issue#</i>	<i>sev/prio</i>	<i>status</i>	<i>description</i>	<i>explanation, status</i>
1.	I/O	COM	stat and st_ino	<p>On UNIX stat.h has:</p> <pre>ino_t st_ino;</pre> <p>but on OpenVMS it has</p> <pre>ino_t st_ino[3];</pre> <p>consequently the st_ino part of the stat structure is a value on UNIX and a pointer on OpenVMS. Generally this means you have to replace something like</p> <pre>if(inode == foo.st_ino)</pre> <p>with</p> <pre>if(inode[0] == foo.st_ino[0] &amp;&amp; inode[1] == foo.st_ino[1] &amp;&amp; inode[2] == foo.st_ino[2])</pre>
2.	I/O	COM	write() to tcp/ip socket	<p>write() is supposed to send as much of a buffer as it can to the output device and then return that value. On OpenVMS write() will fail if it attempts to write more than 64k bytes to a socket. So even though write() is usually used in a loop that cycles through until a buffer is written out it won't work correctly on OpenVMS if the buffer is too big. The workaround is to put in an #ifdef that restricts the transfer to &lt; 64k bytes at one time.</p>
3.	I/O	COM	use of select()	<p>OpenVMS select() only works on sockets, it does NOT work on files. Therefore code which uses select() to synchronize IO for both sockets and files will not function.</p> <p>The fix is reasonably complex. For an example see: <a href="http://seqaxp.bio.caltech.edu/pub/SOFTWARE/XTERM_VMS_122_TOP.ZIP">http://seqaxp.bio.caltech.edu/pub/SOFTWARE/XTERM_VMS_122_TOP.ZIP</a></p>
4.	E/16	CUR	foo.tar.gz	<p>UNIX doesn't care what a file is, OpenVMS does. ODS2 will only accept one ".extension" in a file name. ODS5 will accept pretty much anything that UNIX will.</p> <p>Starting with OpenVMS 7.3-2 even the system disk can be ODS5 formatted. But be careful not all layered products support this yet.</p> <p>(OO) using new CRTL features promisses a (more) simple port when using only ODS5 disks. At this moment we plan to do a ODS5 only port.</p>



<i>issue#</i>	<i>sev/prio</i>	<i>status</i>	<i>description</i>	<i>explanation, status</i>
5.	W/4	COM	A zillion warnings on compilation	<p>Bad news Bucky, most software is crap and you're looking at it. The OpenVMS compilers are much pickier by default than those on other platforms and show you where the problems are. Your best bet in general is to fix the warnings you see, and make the compiler air out all the code's dirty laundry. Get your code to compile cleanly with</p> <pre>/standard=ansi89/prefix=all/warn=enable=(level4,questcode)</pre> <p>and you'll save yourself a lot of trouble later on. It will also make your code more portable.</p> <p>(OO) we know from the porting newsgroup and from direct contact that there will be lots of 64 bit issues and typical compiler version specific workarounds in the code. We'll try to produce clean code, but won't modify general code just for cleanliness.</p>
6.	E/16	CUR	use of UNIX paths	<p>Most UNIX code thinks that it can always take a path and add "/blah/foo.bar" to get a relative path. This will work on OpenVMS as well so long as the initial path is in UNIX format. However if somebody entered: "USRDISK:[JOE.TEST]" then the full file spec will be a hybrid: "USRDISK:[JOE.TEST]/blah/foo.bar" which won't fly. Compaq C provides functions for converting back and forth between OpenVMS and UNIX file/path specs. In general, you want to stick with UNIX file paths since compilers on some other nonUNIX platforms support them, but OpenVMS file paths are only useful on OpenVMS.</p> <p>Ultimate fix: a standard set of C routines which define a file/path structure something like:</p> <pre>char * nodename; struct ACCESSSTRU access; char * device; struct PATHSTRU path; char * name; char * type; char * version;</pre> <p>which will hide all the delimiter information (so no more explicit testing of []:\. etc. in the code). And even then programmers will have to stick to the routines which manipulate this structure. So long as all this information is lumped into one character string inside the code there will always be portability problems. This was a design flaw in the C language which should have been handled back when "fopen" was first defined.</p> <p>(OO) using the new CRTL features promises a (more) simple port when using only ODS5 disks. At this moment we plan to do a ODS5 only port.</p>

<i>issue#</i>	<i>sev/prio</i>	<i>status</i>	<i>description</i>	<i>explanation, status</i>
7.	I/O	LEF	stream-lf and long writes	<p>The default file type produced by the C RTL on OpenVMS is stream-lf. It is very similar to the text file format on UNIX EXCEPT that records can't go above 32767 bytes. Well, they can, but they get split and basically it doesn't quite work like it does on UNIX.</p> <p>Workaround: If possible, use binary file types. Also complain loudly and bitterly (and most likely, futilely) to HP and maybe someday they'll fix it.</p> <p>(OO) With the DECC\$DEFAULT_UDF_RECORD feature switch enabled, file access mode defaults to RECORD instead of STREAM mode for all files except STREAMLF.</p>
8.	I/O	LEF	what about X11?	<p>That's a whole other subject. See: "X11/Motif portability concerns, UNIX to OpenVMS" at <a href="http://seqaxp.bio.caltech.edu/www/X11_VMS_NOTES.TXT">http://seqaxp.bio.caltech.edu/www/X11_VMS_NOTES.TXT</a></p>
9.	W/4	COM	data types	<p>Ironically this will get you most often when you port from Tru64. "long int" there is 64 bits, but it's only 32 bits on OpenVMS. Tru64 also has a long long type which is 128 bits. For "long int" you can usually #ifdef in "unsigned __int64" on the OpenVMS version. For "long long" best hope that the code doesn't really need 128 bits, because you can't get it on this platform!</p> <p>(OO) we know from the porting newsgroup and from direct contact that there will be lots of 64 bit issues and typical compiler version specific workarounds in the code. We'll try to produce clean code, but won't modify general code just for cleanliness.</p>
10.	I/O	LEF	ioctl for terminals	<p>If you encounter ioctl() calls being used to control a terminal you are out of luck. The only solution is to rewrite the code using QIO's. You will usually need to access the documentation from the source UNIX's, as ioctl isn't all that portable among UNIX's either.</p> <p>(OO) POSIX.! deprecated ioctl() a long time ago, because it isn't portable. But the POSIX.! recommended termios.h with tc*() and cf*() functions are not available on OpenVMS either.</p>
11.	E/16	CUR	makefile	<p>OpenVMS has no "make". You can rewrite makefile to descrip.mms and use either MMS (part of DECset) or MMK (free) and obtain similar functionality. Your other choice is to install GNV from <a href="http://gnv.sourceforge.net/">http://gnv.sourceforge.net/</a>. It has bash and gmake.</p> <p>(OO) We know we have to have DMAKE. This MAKE version is the Achilles-heel of this port. The whole project depends on the availability of DMAKE. We have a alpha version working. Continued improvements in the CRTL and the porting library will make DMAKE a more stable and reliable product.</p>
12.	I/O	LEF	lines are too long to view	<p>Many of the OpenVMS tools don't like the wide records which are often found in source code and other files originating on UNIX systems. For instance, you can't TYPE many files, and EDIT/EDT chokes on long records too. I usually use NEDIT <a href="http://www.decus.de:8080/www/vms/sw/nedit.htmlx">http://www.decus.de:8080/www/vms/sw/nedit.htmlx</a> to handle these sorts of file problems.</p>

<i>issue#</i>	<i>sev/prio</i>	<i>status</i>	<i>description</i>	<i>explanation, status</i>
13.	I/O	LEF	why does it run so slowly?	<p>After you complete your port you will often find that the OpenVMS version is much, much slower than the UNIX version if it is very IO intensive. The ratio can be as high as 100X faster for UNIX over OpenVMS. Most often this is due to the file caching speedup from UNIX, plus the 2-6X slowdown that RMS imposes on record oriented files (as measured on a RAMdisk.) Binary IO speeds will be closer, the IO slowdown is primarily a problem for text/record oriented files. You can improve performance by using the SET RMS command to increase the block and buffer sizes. Also increase the /extend value to avoid a flurry of short extends. If you know ahead of time the final size of the output file set extend to that. (You can also set these parameters in the code by using OpenVMS specific extensions.) Improvements in the OpenVMS file caching system are in late stages of development, and those should help once they appear on production systems.</p> <p>That said, watch out for the use of, umm, unusual file techniques on programs from UNIX. For instance, you may find frequent calls to freopen(), or other program methods which cause the disk heads to fly back and forth between the front of the file and the end. These sorts of bad design are masked on UNIX by the file caching system, but are exposed in all their glory on OpenVMS.</p> <p>Programs may also run slowly on OpenVMS, or not at all, if they expect to be able to grab vast amounts of memory. User accounts on OpenVMS are usually allocated a relatively small amount of virtual and physical memory.</p> <p>(OO) The recent OpenVMS ports of Apache 2.0 and Mozilla 1.5 prove that this no longer an issue.</p>
14.	I/O	LEF	why do I get two (or more) copies of the output file?	<p>Many UNIX programs try to determine if they can write an output file by doing an fopen, and if it succeeds, doing another. On UNIX the second one overwrites the first and no one's the wiser. On OpenVMS you get two versions. Sometimes scratch files are used in a similar manner, only for those they tend to be opened and closed many more times, resulting in a long string of files.</p> <p>Fix: find the offending second fopen() and #ifdef it and the fclose() before it out of the code. For scratch files, either reopen the existing file or be sure to delete the file after the fclose().</p>
15.	I/O	LEF	fseek/ftell don't work right	<p>These assume a UNIXlike file organization which may or may not be present on files being read by a C program in OpenVMS.</p> <p>Fix: rewrite the code to use fgetpos() and fsetpos().</p> <p>(OO) Resent changes to the CRTLIB solve this issue.</p>

<i>issue#</i>	<i>sev/prio</i>	<i>status</i>	<i>description</i>	<i>explanation, status</i>
16.	W/4	CUR	linker can't resolve ReadDir	<p>By default UNIX is case sensitive pretty much everywhere. OpenVMS is not. So the symbol "ReadDir" goes to "readdir" which is the standard function. You'll also see this a lot where people have a variable "foobar" and a function "Foobar". It's generally a problem any place code uses case to distinguish between variables and/or functions. (Constants, which only the compiler sees, tend not to cause so much havoc.) Code written this way is really a pain to port because you (usually) didn't write it, and so tend not to notice these minor case differences. The bugs which can result if the linker does manage to resolve things, but incorrectly, can be very messy.</p> <p>Preferred fix: rewrite the code so that all symbols are unique when uppercased.</p> <p>Other fix: use the compiler switch /names=as_is, which preserves case.</p> <p>(OO) Our preferred method is /names=(as_is,short).</p>
17.	I/O	LEF		<p>compiler gives an implicit function warning for a common function</p> <p>This happens either because the appropriate header has been omitted or because the function in question is #ifdef'd off by language standard specific defines. Most UNIX compilers are really sloppy about which function is in which language spec. Ok, they're not sloppy, but the programmers never put them into a mode where they check. So some function which is an XOPEN extension compiles in without warnings on UNIX but raises a warning on OpenVMS.</p> <p>Fix: Use /define=( _XOPEN_SOURCE, _XOPEN_SOURCE_EXTENDED, _POSIX_SOURCE) as required to get the right lines of the header files processed.</p> <p>(OO) Please read the Macro's chapter and The CRTL Reference Guide (1.5.2 Selecting Standard; page 1-19). The macro _XOPEN_SOURCE_EXTENDED already implies _XOPEN_SOURCE and _POSIX_SOURCE!</p>
18.	W/2	COM	where are the standard header files?	<p>They always live in the text library: SYS\$COMMON:[SYSLIB]DECC\$RTLDEF.TLB but may also have been expanded into SYS\$SYSDEVICE:[SYS0.SYSCOMMON.DECC\$LIB.REFERENCE.DECC\$RTLDEF]*.h</p> <p>(OO) Be aware that the HP C/C++ compilers use the text libraries in SYS\$COMMON:[SYSLIB]DECC\$RTLDEF.TLB, and that the text version in SYS\$SYSDEVICE:[SYS0.SYSCOMMON.DECC\$LIB.REFERENCE.DECC\$RTLDEF]*.h are only for reference purposes and may not be up-to-date!</p>

<i>issue#</i>	<i>sev/prio</i>	<i>status</i>	<i>description</i>	<i>explanation, status</i>
19.	E/4	COM	what's wrong with #include <../foo/woo/blah.h>?	<p>Putting paths into includes is BAD, EVIL, YUCKY, etc., and unfortunately, fairly common. The example shown here is the worst case - you'll only ever have a hope of compiling this if the default directory is in exactly the right place.</p> <p>Preferred fix: Whenever possible eliminate the paths from the code and use /include=([-],[],[.foo]) and the like to tell the compiler where to find them. (These are the equivalent of using on UNIX: -I. -I. -Ifoo)</p> <p>Desperate measures fix: For an include like you can define a concealed logical "foo" which points to the correct place in the directory structure on your system and Compaq C will be able to look in the [.woo.moo] subdirectory of it and find the file blah.h.</p> <p>(OO) Although I don't remember ever having a problem with include paths, the current version of the HP C/C++ compilers handles UNIX style include paths very well, except on the command line. The GNV gcc wrapper solves this issue.</p>
20.	I/O	LEF	the compiler includes "foo.h" but not "Foo.h"	<p>OpenVMS is case insensitive. Even ODS5, which will preserve case, will not distinguish between these two files. Usually this shows up long before the compiler gets a shot at it when the archive is unpacked and complains/warns that it is overwriting FOO.H.</p> <p>Fix: rewrite the code to use case invariant file names.</p> <p>(OO) Starting with OpenVMS 7.3-1 you can turn on case sensitive behavior. In DCL you do this with:</p> <div style="border: 1px solid black; padding: 2px; margin-top: 10px;"> \$ SET PROCESS/CASE=SENSITIVE </div>

<i>issue#</i>	<i>sev/prio</i>	<i>status</i>	<i>description</i>	<i>explanation, status</i>
21.	I/O	LEF	how do I find a memory access problem?	<p>Often code which ran "correctly" on one platform will fail miserably on another if it access memory out of bounds. Such "working" code will usually fail when you port it to OpenVMS just because variables will be located in memory differently.</p> <p>One method of dealing with this is to do:</p> <pre> #ifdef __VMS #define free myfree int decc\$free(void *ptr); /* prototype for decc\$free */ void myfree( void *ptr){ #include #include if(decc\$free(ptr) != 0){ (void) printf("illegal free() operation\n"); (void) lib\$signal(SS\$_ACCVIO); } } #endif /* __VMS */ </pre> <p>decc\$free will complain if you try to free a region of memory, whereas free() cannot warn you about the mayhem which is being committed. Many memory errors begin or end with an invalid free() and this will catch them.</p> <p>Here is a method suggested by Hoff (Stephen) Hoffman and posted in comp.os.vms</p> <p>Because of the likelihood of application programming errors -- trampling past the end of the allocated storage being most common -- I almost never call malloc and free directly. Further, by intercepting the memory allocation calls, I can call lib\$get_vm and similar, particularly using VM zones, and can tailor the particular behavior most appropriate. With a "temporary memporary pool" VM zone, I can also flush all allocations in that zone in a single call, so that I can easily reset the pool... Also available with VM zones are the tools to traverse and report on the VM zones.</p> <p>Jacket your calls to malloc and free. Code the malloc jacket to add a quadword at the front of the allocated area and a quadword at the back and then call the actual malloc asking for the requested size plus sixteen bytes. Before returning the allocated storage -- the base address of the allocated block plus eight -- fill the front and back quadword with a known (and no zero bytes in the quadword) pattern, possibly based on the address and size, etc. Code the free jacket to check for the patterns and scream if it finds errors, and to call free if not -- remembering to back up the base of the buffer by eight bytes from what the caller passed in. (I typically refer to these quadwords as "fenceposts" -- straying from the expected behaviour is quite easy to track down.)</p> <p>If you were on more recent OpenVMS Alpha version, you could use the heap analyzer in the OpenVMS Debugger to poke around.</p> <p>(OO) Also note that there are several compiler options to enable or disable things like bounds checking.</p>

<i>issue#</i>	<i>sev/prio</i>	<i>status</i>	<i>description</i>	<i>explanation, status</i>
22.	E/24	CEF	fork()	<p>OpenVMS vfork() doesn't work like fork(). Search your code for fork() and if you find it, refer to gnuplot, or some other already ported application, for an example of how to replace the code.</p> <p>(OO) Also note that OpenVMS vfork() doesn't quite work like UNIX vfork(). See CRTL Reference Guide (Chapter 5 Subprocess Functions).</p>
23.	I/O	LEF	where do I look for help?	<p>Post to the newsgroup comp.os.vms</p> <p>Use dejanews power search at <a href="http://www.dejanews.com/home_ps.shtml">http://www.dejanews.com/home_ps.shtml</a> to search that newsgroup. (Be aware that the default setting only goes back about a year, use the date fields on the bottom to look back further.)</p> <p>"Ask the wizard" at <a href="http://h71000.www7.hp.com/wizard/">http://h71000.www7.hp.com/wizard/</a></p> <p>Get a copy of the freeware CD or other already ported programs and scan them for solutions to the current problem. The freeware CD is on the net at: <a href="http://h71000.www7.hp.com/openvms/freeware/">http://h71000.www7.hp.com/openvms/freeware/</a></p> <p>(OO) The GNV <a href="http://gnv.sourceforge.net/">http://gnv.sourceforge.net/</a> site, the HP Open Source Tools page <a href="http://h71000.www7.hp.com/opensource/opensource.html">http://h71000.www7.hp.com/opensource/opensource.html</a> and our own sites <a href="http://www.4ovms.dyndns.org">http://www.4ovms.dyndns.org</a> and <a href="http://www.oooovms.dyndns.org">http://www.oooovms.dyndns.org</a> to name a few more.</p>
24.	I/O	LEF	system()	<p>system() passes a command line to a subprocess, executes it, and then returns. Since the command line is by definition OS specific, any instances of system() in ported code must inevitably be rewritten. Ideally they should be removed entirely - find some way to do perform the desired action in code rather than by running a subprocess.</p> <p>(OO) It isn't OS specific. It's shell specific. Also note that with GNV installed many UNIX commands work.</p>
25(JE M).	I/O	LEF	getenv() behavior	<p>The behavior of getenv() depends on the version and ECO of the DEC C RTL and possibly logical names defined on your system.</p> <p>Fix: use sys\$trnlnm() instead.</p> <p>(DRM comment: I've also observed instances where symbols and logicals interfered with the expected behavior. This happened primarily when the symbol/logical involved was a "standard" symbol that getenv handled differently. For instance, if "term" was previously defined in DCL than getenv("term") would return the user defined value, which often was not at all the desired quantity, and the program would fail.)</p> <p>(OO) Recent versions of the CRTL may have fixed this issue.</p>

<b>issue#</b>	<b>sev/prio</b>	<b>status</b>	<b>description</b>	<b>explanation, status</b>
26(JE M).	I/O	LEF	mmap() on text files	<p>mmap() will open files in a binary mode. That is fine so long as the file is stream-lf or binary. However, if it is some other type of RMS file the program doing this will likely fail when it encounters the embedded RMS information.</p> <p>Fix (DRM): require input files to be either stream-lf or binary, or if that is not possible, do not use mmap(). Either way will likely require extensive recoding.</p> <p>(OO) Recent versions of the CRTL may have fixed this issue.</p>
27(JE M).	I/O	LEF	/tmp and others map to locations, overridden by logical names	<p>Apparently some versions of the C RTL map /tmp to SYS\$SCRATCH /dev/null to NLA0:, and perhaps some others are translated as well. In the case of /tmp if you have defined a logical TMP and try to write to /tmp/foo.txt it will go to the directory set by the logical. This is consistent with the usage of logicals for paths which are not "special". Just be aware that the logical TMP or DEV may have been defined with some other meaning, so that /dev/null might end up creating a real file NULL in the directory pointed to by the logical DEV!</p> <p>(OO) There are several CRTL feature switches to tune this behavior.</p>
28(JE M).	I/O	LEF	setuid()	<p>The setuid() function is a stub and just returns 0 to indicate "success". That may not be optimal, since it really failed (the UID did not change.) User written setuid() is possible, but when the transfer is to an account which does not have read access to the calling processes job table, then SYS\$SCRATCH and SYS\$LOGIN will not be available.</p> <p>(OO) Starting with OpenVMS 7.3-2 you can enable POSIX style ID's, with the DECC\$POSIX_STYLE_UID feature switch. With POSIX style ID's enabled setuid() is fully functional.</p>
29(JE M).	I/O	LEF	uid_t/gid_t size inconsistencies	<p>getuid() returns 16 bits (the member part of a UIC) with some compiler flags or on VMS 6.x and under, and 32 bits (the whole UIC) otherwise. stat() returns a 32 bit uid and a 16 bit gid. setuid() (user written) will need a 32 bit uid value.</p> <p>(OO) For some time now, you can compile with the macro __USE_LONG_GID_T to enable 32 bit gid and uid values.</p>



<i>issue#</i>	<i>sev/prio</i>	<i>status</i>	<i>description</i>	<i>explanation, status</i>
30.	I/O	LEF	Files produced by the C RTL have invalid RMS attributes	<p>This isn't a portability problem per se, but rather a general C RTL problem which will affects all ported programs. The problem is that sequential text files written by the C RTL default to having a Longest Record value of 32767, and this can cause havoc with some other programs which foolishly believe that the value is accurate. For instance, SORT will allocate 32767 bytes/record for any such file, so that it cannot easily sort even a relatively small file. ( The work around in this instance is to either use SORT/PROCESS=tag or to figure out how big LRL is and repair the file with SET FILE/ATTRIB=(lrl:whatever) before sorting it.)</p> <p>Partial fix: (This is really a sad excuse for a solution but it's all there is)</p> <pre>\$ define/user decc\$default_lrl 100</pre> <p>will set the LRL value to 100 instead of 32767. However, the 100 can be just as wrong as the 32767 - write a record of length 16k into that file and the RMS value will stay at 100. The C RTL should really just keep track of the records it writes and put the REAL value.</p> <p>Section 1.6 of the Compaq C Run-Time Library Reference Manual is the only place this logical is mentioned:</p> <p>In OpenVMS Version 7.0 the default LRL value on stream files was changed from 0 to 32767. This change caused significant performance degradation on certain file operations such as sort.</p> <p>This is no longer a problem. The Compaq C Run-Time Library now lets you define the logical DECC\$DEFAULT_LRL to change the default record-length value on stream files.</p> <p>The Compaq C Run-Time Library first looks for this logical. If it is found and it translates to a numeric value between 0 and 32767, that value is used for the default LRL.</p> <p>I disagree emphatically that "this is no longer a problem". The value of LRL set by DECC\$DEFAULT_LRL is every bit as wrong as the default LRL. Its only advantage is that it can be "small and wrong" rather than "large and wrong."</p>

<i>issue#</i>	<i>sev/prio</i>	<i>status</i>	<i>description</i>	<i>explanation, status</i>
31.	I/O	LEF	Where do I obtain tools and libraries for porting?	<p>bash,gmake, and many others are part of GNV  <a href="http://gnv.sourceforge.net/">http://gnv.sourceforge.net/</a></p> <p>qt 2.1 <a href="http://www.lehrig.de/service/service2.php">http://www.lehrig.de/service/service2.php</a></p> <p>xforms 0.88 (warning, bugs in some versions of the C RTL and some graphics drivers are triggered by this software! Includes a port of XPM in shareable format)  <a href="http://world.std.com/~xforms/">http://world.std.com/~xforms/</a></p> <p>parallel programming: pvm, tcgmsg (maybe MPICH too, but it wasn't there as this was written)  <a href="ftp://v36.chemie.uni-konstanz.de/">ftp://v36.chemie.uni-konstanz.de/</a></p> <p>OpenVMS Porting Library from Compaq (September 2003)  <a href="http://h71000.www7.hp.com/openvms/products/ips/porting.html">http://h71000.www7.hp.com/openvms/products/ips/porting.html</a></p> <p>Many libraries may be found in already completed ports see:  <a href="http://h71000.www7.hp.com/openvms/freeware/index.html">http://h71000.www7.hp.com/openvms/freeware/index.html</a>  <a href="http://h71000.www7.hp.com/freeware/">http://h71000.www7.hp.com/freeware/</a></p> <p>Use advanced search from <a href="http://groups.google.com/">http://groups.google.com/</a> to look for posts in comp.os.vms concerning the package of interest.</p> <p>(OO) The HP Open Source Tools page  <a href="http://h71000.www7.hp.com/opensource/opensource.html">http://h71000.www7.hp.com/opensource/opensource.html</a>, The HP e-business technology page  <a href="http://h71000.www7.hp.com/ebusiness/technology.html">http://h71000.www7.hp.com/ebusiness/technology.html</a>,  Jouk Jansen's OpenVMS page  <a href="http://nchrem.tnw.tudelft.nl/openvms/software2.html">http://nchrem.tnw.tudelft.nl/openvms/software2.html</a> and  our own sites <a href="http://www.4ovms.dyndns.org">http://www.4ovms.dyndns.org</a> and  <a href="http://www.oooovms.dyndns.org">http://www.oooovms.dyndns.org</a> to name a few more.</p>

## Example 1: m4

In this example I use GNU m4 1.4. By the way as you can see below I use POSIX.2 PAX to untar the tar file.

```
$ set process /parse_style=extended
$ gunzip m4-1.4.tar.gz
$ pax -rvf m4-1.4.tar
USTAR format archive assumed
m4-1.4/README
m4-1.4/NEWS
m4-1.4/TODO
m4-1.4/THANKS
m4-1.4/COPYING
m4-1.4/INSTALL
.
.
.
```

Let me not bore you with a long list of files.

Before I start bash I'll raise DECC\$PIPE\_BUFFER\_SIZE:

```
$ define/job decc$pipe_buffer_size 65000
```

Start Bash:

```
$ bash
```

Disable DCL fallback:

```
bash$ export GNV_DISABLE_DCL_FALLBACK=1
bash$ ls
m4-1.4  m4-1.4.sav  m4-1.4.tar
bash$ cd m4-1.4
bash$ ls
BACKLOG      Makefile.in  TODO        checks      doc          mkinstalldirs
COPYING      NEWS         acconfig.h  config.h.in examples     src
ChangeLog    README       aclocal.m4  configure   install-sh   stamp-h.in
INSTALL      THANKS       c-boxes.el  configure.in lib
```

let's start configure with following options:

`--build=build'`

Specifies the type of system on which the package will be built. If not specified, the default will be the same configuration name as the host.

The parameter for the build option is in the following form:

`<processor>-<OS Manufacturer>-<OS name>`

Note that we don't have to set the execute bit to start the configure script.

```
bash$ ./configure --build=alpha-hp-vms
creating cache ./config.cache
checking for mawk... no
checking for gawk... gawk
checking for gcc... gcc
.
.
.
creating Makefile
```

```

creating doc/Makefile
creating lib/Makefile
creating src/Makefile
creating checks/Makefile
creating examples/Makefile
creating config.h

```

Wow, no problems so far. Let's try make.

```

bash$ make
for subdir in doc lib src checks examples; do \
  echo making all in $subdir; \
  (cd $subdir && make CC='gcc' CFLAGS='-g' LDFLAGS='' LIBS=''
prefix='/usr/local' exec_prefix='/usr/local' bindir='/usr/local/bin'
infodir='/usr/local/info' all) || exit 1; \
done
making all in doc
make[1]: Entering directory `/USER50/KITS/GNU/M4/m4-1.4/doc'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/USER50/KITS/GNU/M4/m4-1.4/doc'
making all in lib
make[1]: Entering directory `/USER50/KITS/GNU/M4/m4-1.4/lib'
gcc -c -DHAVE_CONFIG_H -I.. -I. -g regex.c
gcc -c -DHAVE_CONFIG_H -I.. -I. -g getopt.c

      if (optind != argc && !strcmp (argv[optind], "--"))
      .....^
%CC-I-IMPLICITFUNC, In this statement, the identifier "strcmp" is implicitly
declared as a function.
at line number 408 in file USER50:[KITS.GNU.M4.m4-1^4.lib]getopt.c;1

      if (!strcmp (p->name, nextchar, nameend - nextchar))
      .....^
%CC-I-IMPLICITFUNC, In this statement, the identifier "strcmp" is implicitly
declared as a function.
at line number 484 in file USER50:[KITS.GNU.M4.m4-1^4.lib]getopt.c;1

      if (nameend - nextchar == strlen (p->name))
      .....^
%CC-I-IMPLICITFUNC, In this statement, the identifier "strlen" is implicitly
declared as a function.
at line number 486 in file USER50:[KITS.GNU.M4.m4-1^4.lib]getopt.c;1
gcc -c -DHAVE_CONFIG_H -I.. -I. -g getopt1.c
gcc -c -DHAVE_CONFIG_H -I.. -I. -g error.c

char *strerror ();
.....^
%CC-W-FUNCREDECL, In this declaration, function types differ because one has no
argument information and the other has an ellipsis.
at line number 56 in file USER50:[KITS.GNU.M4.m4-1^4.lib]error.c;1
gcc -c -DHAVE_CONFIG_H -I.. -I. -g obstack.c

      abort ();
      ....^
%CC-I-IMPLICITFUNC, In this statement, the identifier "abort" is implicitly
declared as a function.
at line number 333 in file USER50:[KITS.GNU.M4.m4-1^4.lib]obstack.c;1
gcc -c -DHAVE_CONFIG_H -I.. -I. -g xmalloc.c
gcc -c -DHAVE_CONFIG_H -I.. -I. -g xstrdup.c
rm -f libm4.a
ar cru libm4.a regex.o getopt.o getopt1.o error.o obstack.o xmalloc.o xstrdup.o
Warning: u unrecognized switch
%LIBRAR-W-COMCOD, compilation warnings in module ERROR file
USER50:[KITS.GNU.M4.m4-1^4.lib]error.o;1
: libm4.a
make[1]: Leaving directory `/USER50/KITS/GNU/M4/m4-1.4/lib'
making all in src
make[1]: Entering directory `/USER50/KITS/GNU/M4/m4-1.4/src'

```

```

gcc -c -DHAVE_CONFIG_H -I.. -I. -I./../lib -g m4.c
gcc -c -DHAVE_CONFIG_H -I.. -I. -I./../lib -g builtin.c
gcc -c -DHAVE_CONFIG_H -I.. -I. -I./../lib -g debug.c
gcc -c -DHAVE_CONFIG_H -I.. -I. -I./../lib -g eval.c
gcc -c -DHAVE_CONFIG_H -I.. -I. -I./../lib -g format.c
gcc -c -DHAVE_CONFIG_H -I.. -I. -I./../lib -g freeze.c
gcc -c -DHAVE_CONFIG_H -I.. -I. -I./../lib -g input.c
gcc -c -DHAVE_CONFIG_H -I.. -I. -I./../lib -g macro.c
gcc -c -DHAVE_CONFIG_H -I.. -I. -I./../lib -g output.c
gcc -c -DHAVE_CONFIG_H -I.. -I. -I./../lib -g path.c
gcc -c -DHAVE_CONFIG_H -I.. -I. -I./../lib -g symtab.c
gcc -o m4 m4.o builtin.o debug.o eval.o format.o freeze.o input.o macro.o
output.o path.o symtab.o ../lib/libm4.a
%LINK-W-WRNNERS, compilation warnings
      in module ERROR file USER50:[KITS.GNU.M4.m4-1^4.lib]libm4.a;1
%LINK-W-MULDEF, symbol DECC$GETOPT multiply defined
      in module DECC$SHR_EV56 file SYS$COMMON:[SYSLIB]DECC$SHR_EV56.EXE;1
%LINK-W-MULDEF, symbol DECC$GA OPTARG multiply defined
      in module DECC$SHR_EV56 file SYS$COMMON:[SYSLIB]DECC$SHR_EV56.EXE;1
%LINK-W-MULDEF, symbol DECC$GL OPTOPT multiply defined
      in module DECC$SHR_EV56 file SYS$COMMON:[SYSLIB]DECC$SHR_EV56.EXE;1
%LINK-W-MULDEF, symbol DECC$GL OPTIND multiply defined
      in module DECC$SHR_EV56 file SYS$COMMON:[SYSLIB]DECC$SHR_EV56.EXE;1
%LINK-W-MULDEF, symbol DECC$GL OPTERR multiply defined
      in module DECC$SHR_EV56 file SYS$COMMON:[SYSLIB]DECC$SHR_EV56.EXE;1
make[1]: Leaving directory `/USER50/KITS/GNU/M4/m4-1.4/src'
making all in checks
make[1]: Entering directory `/USER50/KITS/GNU/M4/m4-1.4/checks'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/USER50/KITS/GNU/M4/m4-1.4/checks'
making all in examples
make[1]: Entering directory `/USER50/KITS/GNU/M4/m4-1.4/examples'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/USER50/KITS/GNU/M4/m4-1.4/examples'

```

OK, we did see some warnings, but nothing too bad. let's try the m4 executable.

```

bash$ cd src
bash$ m4 --help
Usage: m4 [OPTION]... [FILE]...
Mandatory or optional arguments to long options are mandatory or optional
for short options too.

Operation modes:
  --help                display this help and exit
  --version             output version information and exit
  -e, --interactive     unbuffer output, ignore interrupts
  -E, --fatal-warnings  stop execution after first warning
  -Q, --quiet, --silent suppress some warnings for builtins
  -P, --prefix-builtins force a `m4_' prefix to all builtins

Preprocessor features:
  -I, --include=DIRECTORY search this directory second for includes
  -D, --define=NAME[=VALUE] enter NAME has having VALUE, or empty
  -U, --undefine=NAME      delete builtin NAME
  -s, --synclines         generate `#line NO "FILE"' lines

Limits control:
  -G, --traditional      suppress all GNU extensions
  -H, --hashsize=PRIME   set symbol lookup hash table size
  -L, --nesting-limit=NUMBER change artificial nesting limit

Frozen state files:
  -F, --freeze-state=FILE produce a frozen state on FILE at end
  -R, --reload-state=FILE reload a frozen state from FILE at start

Debugging:
  -d, --debug=[FLAGS]    set debug level (no FLAGS implies `aeq')

```

```
-t, --trace=NAME          trace NAME when it will be defined
-l, --arglength=NUM       restrict macro tracing size
-o, --error-output=FILE   redirect debug and trace output
```

FLAGS is any of:

```
t   trace for all macro calls, not only traceon'ed
a   show actual arguments
e   show expansion
q   quote values as necessary, with a or e flag
c   show before collect, after collect and after call
x   add a unique macro call id, useful with c flag
f   say current input file name
l   say current input line number
p   show results of path searches
i   show changes in input files
V   shorthand for all of the above flags
```

If no FILE or if FILE is '-', standard input is read.

```
bash$
```

And it works!?!

But we are far from ready.

- we need to set the necessary feature switches by linking a LIB\$INITIALIZE routine to our program
- the CRTL feature switches are not the only way to make some functions behave more UNIX like. For example you can define `_POSIX_EXIT` to make the CRTL exit function behave POSIX compliant. Please note that the `_POSIX_EXIT` macro is already defined by the gcc wrapper!
- As an OpenVMS person, I really can't live with all these warnings

To fix the LIB\$INITIALIZE problem, you could do the following (you're still in the src directory):

```
bash$ cp /src/GNV/CRTLSUP/SRC/VMS_CRTL_INIT.C .
```

Edit the Makefile in this directory to add the VMS\_CRTL\_INIT.C source file and VMS\_CRTL\_INIT.o object file to the following lines:

```
SOURCES = m4.c builtin.c debug.c eval.c format.c freeze.c input.c \
macro.c output.c path.c stackovf.c symtab.c
OBJECTS = m4$O builtin$O debug$O eval$O format$O freeze$O input$O \
macro$O output$O path$O $(STACKOVF) symtab$O
```

After the changes the lines should look like this:

```
SOURCES = m4.c builtin.c debug.c eval.c format.c freeze.c input.c \
macro.c output.c path.c stackovf.c symtab.c VMS_CRTL_INIT.C
OBJECTS = m4$O builtin$O debug$O eval$O format$O freeze$O input$O \
macro$O output$O path$O $(STACKOVF) symtab$O VMS_CRTL_INIT$O
```

To fix the compile warnings you'll need to make some changes to various files. Start with config.h in the root directory of the package. When you look at this file, you'll understand why.

The linker warnings are the result of the fact that the m4 sources contain replacement functions for the standard command-line argument parsing functions with the same name's as the standard functions. You can either disable these replacements or do something like the following:

Go to the end of the config.h file and add the following lines:

```
#define getopt my_getopt
#define optarg my_optarg
#define optopt my_optopt
#define optind my_optind
#define opterr my_opterr
```

Now run `./configure` and make again.

So in short:

```
bash$ make clean
bash$ make
```

GNU m4 also comes with some scripts to check its functionality so let's take a look how well it does.

```
bash$ make check
.
.
.
cd checks && make check
make[1]: Entering directory `/USER50/KITS/GNU/M4/m4-1.4/checks'
PATH=`pwd`/../../src:$PATH; export PATH; \
cd . && ./check-them *[0-9][0-9].*
GNU m4 1.4
Checking 01.define
.
.
.
Checking 30.include
../doc/m4.texinfo:2078: Origin of test
30.include: stderr mismatch
lcl
< 30.include:2: m4: Cannot open no-such-file: No such file or directory
---
> 30.include:2: m4: Cannot open no-such-file: no such file or directory
Checking 31.include
.
.
.
Checking 57.m4exit

Failed checks were:
  30.include:err
make[1]: Leaving directory `/USER50/KITS/GNU/M4/m4-1.4/checks'
bash$
```

The test that failed is only complaining about the error message not being exactly right.

All in all this looks quite good already. But things aren't always this easy.