

CENTERLINE-C++ PLATFORM GUIDE FOR HP USERS VERSION 2.1.1

HP SYSTEM REQUIREMENTS

For CenterLine-C++'s memory, swap space, and disk space requirements, please see the Release Bulletin that accompanies this release.

Supported platforms

CenterLine-C++ Version 2.1.1 supports HP 9000 Series 700 workstations running HP-UX 9.0.1, 9.0.3, and 9.0.5 and Series 800 workstations running HP-UX 9.0.

The process debugging mode (pdm) used in this version of CenterLine-C++ is based on GNU gdb, Version 4.13.

To license its software, this version of CenterLine-C++ uses FLEXlm, Version 2.40c.

Supported compilers

CenterLine-C++ supports the following C compilers on the HP platform:

- o CenterLine-C compiler (clcc)
- o HP92453-01 A.09.61 HP C Compilers (cc and c89)

CenterLine-C++ and CenterLine-C are link-compatible with the following compilers:

- o FORTRAN
- o HP C++ compilers (as long as object modules are not compiled with the +eh switch)

This version of CenterLine-C++ supports Release 3.0.2.15 of the USL C++ Language System, which is backward compatible with Release 2.x and 3.0.x. We support everything in the USL C++ release, except that we do not support or supply the task library (libtask.a).

In most cases, code that compiled without warning in USL C++ Release 2.x or 3.0.x will compile correctly in Release 3.0.2.15. However, you may run across some cases that cause problems. For instance, you may find that some cases of switch statements fail to compile, complaining about jumping past an initializer; the fix here is to supply curly braces ({}), around the body of the case statement. For best results, put the break statement, if any, outside the closing curly brace.

You may also encounter an "unresolved symbol" problem at link time if you have link symbols that contain nested types, including nested enums. USL C++ Release 2.x, 3.0.x, and 3.0.2.15 encode the names of such symbols differently. The fix is to recompile the problem modules with cfront 3.0.2.15.

Fortunately, this situation is uncommon.

Supported windowing systems

CenterLine-C++ supports both the Motif and OPEN LOOK windowing systems. Motif is the default on the HP platform. You can choose OPEN LOOK at startup with the `-openlook` switch on the `centerline-c++` command line.

POTENTIAL HP ANOMALIES

There are some platform-specific features that CenterLine-C++ may not support fully, so you may see unexpected behavior. This entry attempts to call your attention to these potential anomalies.

No libc.a on HP platforms

We do not provide the CenterLine-C ANSI C library on HP platforms because the standard C library distributed with HP-UX is ANSI C compliant.

Static symbol not in symtab

On the HP platform, `pdm` may issue the following warning:

```
Internal: static symbol `symname' found in filename psyntab but not in syntab
```

If you receive this message, issue the following command:

```
whatis <symname>
```

and then reissue the command that caused the warning.

Attaching to a running process may fail on HP-UX

Attaching to a running process sometimes fails on HP-UX in CenterLine-C++ if the `pdm` binary you are using is installed on a remote partition.

The failure looks like a `ptrace` failure, such as one you get if you request an illegal process number or if you attempt to attach a file you do not own.

```
pdm -> debug a.out
process_id debug: ptrace: Permission denied.
pdm ->
```

Before implementing the workaround we provide below, eliminate any possibility that the failure is the result of a `ptrace` error. See the `ptrace` man page for information about `ptrace` failure errors.

This is the workaround:

- o Copy the `pdm` binary to a disk that is local to the HP-UX machine.

You must copy it on a disk local to each HP-UX

machine on which you want pdm to run. Do not install it on a cluster partition.

- o Create a .clpm.conf file in your home directory to override the AS PDM directive in the system clpm.conf.

If you have multiple home directories, you have to create multiple ~/.clpm.conf files so that the workaround affects each machine on which you want pdm to run.

The clpm.conf file should contain the following lines.

```
AS PDM {
  BINARY: /tmp/pdm;
  BINARY_ENV: CLDB;
  ENV:
  LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${centerline}/${arch}/lib;
  ARGS: -connect;
  RESTART_ARGS: -connect -restart;
  ARG_TMPL: ^[^-].*:0;
}
```

With the workaround, attempts to attach to a running process will now succeed:

```
pdm 1 -> debug a.out pid
Debugging program `/net/pickup/tmp/loop'
Resetting to top level.
warning: reading register r4: I/O error
0x2560 in sigpause ()
pdm (break 1) 2 -> where
#0 0x2560 in sigpause ()
#1 0x21cc in _sleep ()
#2 0x1fec in main () at loop.c:16
```

Attaching to a process may also fail if the process is sleeping. pdm cannot attach to a sleeping process.

For more information about attaching to a running process, see the debug entry in the Manual Browser or CenterLine-C++ Programmer's Guide and Reference.

Shared library stack frames in core files

pdm is currently unable to report about shared library stack frames when statically analyzing core files. Instead, the output will look something like this:

```
Core was generated by `a.out'.
Program terminated with signal 6, Aborted.
You can't do that without a process to debug
#0 0x800ab0c8 in _end ()
pdm 1 -> where
#0 0x800ab0c8 in _end ()
#1 0x800ab098 in _end ()
```

In order to get an accurate stack trace, reproduce the fault within pdm.

```
pdm 1 -> run
Resetting to top level.
Executing: a.out

Reading symbols from /lib/libc.sl...no debugging symbols found...done.

Program received signal SIGABRT, Aborted.
0x7b0110c8 in kill ()
pdm (break 1) 2 -> where
#0 0x7b0110c8 in kill ()
#1 0x7b011098 in export stub
#2 0x7aff3368 in raise ()
#3 0x7aff32e4 in export stub
#4 0x7aff2f18 in abort ()
#5 0x7aff2d34 in export stub
#6 0x1e74 in foo(const void *, const void *) (a=0x0, b=0xa "") at h.c:5
#7 0x1e10 in export stub
#8 0x7afe3a84 in _qsort ()
#9 0x7afe39ac in _qsort ()
#10 0x7afe38d4 in export stub
#11 0x1ed4 in main ()
```

Link with libCdynamic to invoke static initializers

The CenterLine C++ translator distributed with CenterLine-C++ 2.1.1 invokes static/global constructors and destructors from shared libraries when linked with a dynamic version of libC. To link with the dynamic library, use the `-set_lib_id` switch on the CC command line:

```
% CC -set_lib_id=Cdynamic file.c
```

Alternatively, you can set the LIB_ID environment variable to Cdynamic before compiling with CC:

```
% setenv LIB_ID Cdynamic
% CC file.c
```

clcc command-line switches

Many of the switches used with the HP C compiler are also used with clcc, the CenterLine-C compiler. However, some switches you use with cc must be replaced by a corresponding clcc switch, and other cc switches have no equivalent in clcc.

The following table shows some common cc switches and their clcc equivalents.

HP cc switch	clcc switch	Description
-Aa	-ansi	Enables strict ANSI compliance
-Ac	-traditional,	Disables strict ANSI C compliance.

-Xa, or -Xt		Please refer to the descriptions of these switches in the CenterLine-C Programmer's Guide or the clcc manual page for more information
-G	-pg	Inserts information required by the gprof profiler in the object file.
-Wx, arg1[, arg2, ..., argn]	-Hcppopt=string -Hldopt=string	Pass argument string to the preprocessor (-Hcppopt) or the linker (-Hldopt). string can contain multiple arguments separated by commas.
+L	-Hlist	Generates a source listing on standard output.
+On	-On	Sets the optimization level to n where n is 1 to 7. If n is not specified, then a level of 2 is used.
+wn	-wn	Suppress warning messages at level n and higher where n is 1 to 4.
+z	-pic	Produces position-independent code. The memory allocated to static variables cannot exceed 4K.
+Z	-PIC	Like -pic, but allows the global offset table to span the range of 32-bit addresses when there are too many global objects for -pic.
