

# \$SPAD/lsp Makefile

The Axiom Team

July 29, 2014

**Abstract**

# Contents

<b>1</b>	<b>The Makefile</b>	<b>3</b>
<b>2</b>	<b>Gnu Common Lisp 2.6.7</b>	<b>3</b>
<b>3</b>	<b>Gnu Common Lisp 2.6.7pre</b>	<b>3</b>
3.1	run-process patch . . . . .	3
<b>4</b>	<b>Gnu Common Lisp 2.6.6</b>	<b>3</b>
4.1	run-process patch . . . . .	3
<b>5</b>	<b>Gnu Common Lisp 2.6.5w</b>	<b>4</b>
5.1	mingw.defs . . . . .	4
5.2	alloc.c . . . . .	4
5.3	mingfile.c . . . . .	4
5.4	unixfsys.c . . . . .	4
<b>6</b>	<b>Gnu Common Lisp 2.6.5</b>	<b>5</b>
6.1	gmp wrappers patch . . . . .	5
<b>7</b>	<b>Gnu Common Lisp 2.5.2</b>	<b>5</b>
7.0.1	socket patch . . . . .	5
7.0.2	read.d patch . . . . .	9
7.0.3	fortran patch . . . . .	9
7.0.4	libspad patch . . . . .	10
7.0.5	toploop patch . . . . .	12
7.0.6	object to float patch . . . . .	14
7.0.7	in-package patch . . . . .	15
7.0.8	EXIT and MAX_STACK_SIZE patches . . . . .	15
7.0.9	tail-recursive patch . . . . .	16
7.0.10	collectfn fix . . . . .	17
7.1	The GCL-2.5.2 stanza . . . . .	21
7.1.1	Configure and Make GCL . . . . .	21
7.2	The GCL-2.6.1 stanza . . . . .	23
7.3	The GCL-2.6.2 stanza . . . . .	24
7.4	Directory move . . . . .	25
7.5	The GCL-2.6.2a stanza . . . . .	25
7.6	Directory move . . . . .	26
7.7	The GCL-2.6.3 stanza . . . . .	26
7.8	The GCL-2.6.5 stanza . . . . .	27
7.9	The GCL-2.6.5w stanza . . . . .	28
7.10	The GCL-2.6.6 stanza . . . . .	29
7.11	The GCL-2.6.7pre stanza . . . . .	30
7.12	The GCL-2.6.7 stanza . . . . .	31
7.13	The GCL-2.6.8pre stanza . . . . .	32
7.14	The GCL-2.6.8pre2 stanza . . . . .	33

7.15	The GCL-2.6.8pre3 stanza . . . . .	34
7.16	The GCL-2.6.8pre4 stanza . . . . .	35
7.17	The GCL-2.6.8pre7 stanza . . . . .	36
<b>8</b>	<b>Gnu Common Lisp 2.5</b>	<b>37</b>
8.0.1	socket patch . . . . .	37
8.0.2	fortran patch . . . . .	37
8.0.3	libspad patch . . . . .	38
8.0.4	toploop patch . . . . .	38
8.0.5	xdrfun bug patch . . . . .	38
8.1	The GCL-2.5 stanza . . . . .	38
<b>9</b>	<b>Gnu Common Lisp 2.4.1 (The default build)</b>	<b>39</b>
9.0.1	socket patch . . . . .	39
9.0.2	fortran patch . . . . .	40
9.0.3	libspad patch . . . . .	40
9.0.4	toploop patch . . . . .	40
<b>10</b>	<b>The Makefile</b>	<b>41</b>
10.1	GCL already installed . . . . .	41
10.2	The GCL-cygwin stanza . . . . .	43

## 1 The Makefile

We create a dummy file **gcl** after gcl has been built so it is not rebuilt. We need to do this because we have no control over the gcl Makefiles.

## 2 Gnu Common Lisp 2.6.7

There is a typo in configure.in that is only detected under some versions of bash. The problem is a missing single-quote mark.

— **gcl-2.6.7.configure.in.patch** —

```
@(cd ${GCLVERSION} ; \  
  echo 28a applying configure.in patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.configure.in.patch )
```

---

## 3 Gnu Common Lisp 2.6.7pre

### 3.1 run-process patch

The gcl-2.6.6.h.linux.h.patch has been accepted into the mainline lisp code and is no longer needed.

## 4 Gnu Common Lisp 2.6.6

We need run-process to handle the new browser and graphics direction.

### 4.1 run-process patch

— **gcl-2.6.6.h.linux.h.patch** —

```
@(cd ${GCLVERSION}/h ; \  
  echo 28a applying run-process patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.h.patch )
```

---

This patch fixes the namestring argument to handle windows-style names with spaces. This patch is no longer needed as this was merged into the CVS HEAD for 2.6.6.

— **gcl-2.6.6.cmpnew.gcl\_cmpmain.lsp.patch** —

```
@(cd ${GCLVERSION}/h ; \  
  echo 1 applying cmpnew/gcl_cmpmain.lsp.patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpmain.lsp.patch )
```

---

## 5 Gnu Common Lisp 2.6.5w

This is a Windows port of GCL. We run under MSYS and have to make a few Windows specific changes.

### 5.1 mingw.defs

This patch adds the necessary .o files to the EXTRAS variable so they are available at link time.

— **gcl-2.6.5w.h.mingw.defs.patch** —

```
@(cd ${GCLVERSION}/h ; \  
  echo 1 applying gcl-2.6.5.h.mingw.defs.patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.mingw.defs.patch )
```

---

### 5.2 alloc.c

If malloc() gets called by the C runtime before main starts and the shared memory is not yet initialised causing failure. We set SET\_REAL\_MAXPAGE which forces a call to init\_shared\_memory().

— **gcl-2.6.5w.o.alloc.c.patch** —

```
@(cd ${GCLVERSION}/o ; \  
  echo 2 applying gcl-2.6.5w.o.alloc.c.patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.o.alloc.c.patch )
```

---

### 5.3 mingfile.c

We do not need to call truename. In fact, truename does not seem to be expanding pathnames properly.

— **gcl-2.6.5w.o.mingfile.c.patch** —

```
@(cd ${GCLVERSION}/o ; \  
  echo 3 applying gcl-2.6.5w.o.mingfile.c.patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.o.mingfile.c.patch )
```

---

### 5.4 unixfsys.c

We have to do conversions of Windows filenames.

— **gcl-2.6.5w.o.unixfsys.c.patch** —

```
@(cd ${GCLVERSION}/o ; \  
  echo 4 applying gcl-2.6.5w.o.unixfsys.c.patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.o.unixfsys.c.patch )
```

---

## 6 Gnu Common Lisp 2.6.5

### 6.1 gmp wrappers patch

The file `gmp_wrappers.h` has the declaration of the integer  $j$  out of order. This causes the compiler to complain. This patch puts the declaration at the beginning of the function.

— **gcl-2.6.5.h.gmp\_wrappers.h.patch** —

```
@(cd ${GCLVERSION}/h ; \  
  echo 28 applying gmp_wrappers patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.gmp_wrappers.h.patch )
```

—————

— **gcl-2.6.5w.h.gmp\_wrappers.h.patch** —

```
@(cd ${GCLVERSION}/h ; \  
  echo 6 applying gmp_wrappers patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.gmp_wrappers.h.patch )
```

—————

## 7 Gnu Common Lisp 2.5.2

### 7.0.1 socket patch

This patch is to `h/386-linux.defs` to include two global variables, **EXTRAS** and **EXTRA\_LIB**

```
EXTRAS = ${OBJ}/${SYS}/lib/cfuns-c.o ${OBJ}/${SYS}/lib/sockio-c.o  
EXTRA_LIB=${OBJ}/${SYS}/lib/libspad.a
```

The **EXTRAS** variable is used to include two files into the running image. The **cfuns-c** file contains low-level directory manipulation routines. The **sockio-c** file contains low level socket code. Note that versions of GCL beyond gcl-2.4.1 may have routines already available. If so the Axiom references to these routines should be rewritten.

We also need to create two `.ini` files, one for `cfuns-c` and one for `sockio-c`. These are referenced in the gcl-2.4.1 makefiles but, since no initialization is needed, we simply create empty files.

— **gcl-2.5.2.socket.patch** —

```
@(cd ${GCLVERSION}/h ; \  
  echo 3 applying EXTRAS patch to h/linux.defs ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )  
@(echo 4 setup ini files for EXTRAS patch ; \  
  touch ${OBJ}/${SYS}/lib/cfuns-c.ini ; \  
  touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

—————

— gcl-2.6.1.socket.patch —

```
@(cd ${GCLVERSION}/h ; \  
  echo 3 applying EXTRAS patch to h/linux.defs ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )  
@(echo 4 setup ini files for EXTRAS patch ; \  
  touch ${OBJ}/${SYS}/lib/cfun-c.ini ; \  
  touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

—————

— gcl-2.6.2.socket.patch —

```
@(cd ${GCLVERSION}/h ; \  
  echo 3 applying EXTRAS patch to h/linux.defs ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )  
@(echo 4 setup ini files for EXTRAS patch ; \  
  touch ${OBJ}/${SYS}/lib/cfun-c.ini ; \  
  touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

—————

— gcl-2.6.2a.socket.patch —

```
@(cd ${GCLVERSION}/h ; \  
  echo 3 applying EXTRAS patch to h/linux.defs ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )  
@(echo 4 setup ini files for EXTRAS patch ; \  
  touch ${OBJ}/${SYS}/lib/cfun-c.ini ; \  
  touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

—————

— gcl-2.6.3.socket.patch —

```
@(cd ${GCLVERSION}/h ; \  
  echo 3 applying EXTRAS patch to h/linux.defs ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )  
@(echo 4 setup ini files for EXTRAS patch ; \  
  touch ${OBJ}/${SYS}/lib/cfun-c.ini ; \  
  touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

—————

— gcl-2.6.5.socket.patch —

```
@(cd ${GCLVERSION}/h ; \  
  echo 3 applying EXTRAS patch to h/linux.defs ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )
```

```
@(echo 4 setup ini files for EXTRAS patch ; \
touch ${OBJ}/${SYS}/lib/cfuncs-c.ini ; \
touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

—————

#### — gcl-2.6.5w.socket.patch —

```
@(cd ${GCLVERSION}/h ; \
echo 19 applying EXTRAS patch to h/linux.defs ; \
${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )
@(echo 20 setup ini files for EXTRAS patch ; \
touch ${OBJ}/${SYS}/lib/cfuncs-c.ini ; \
touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

—————

#### — gcl-2.6.6.socket.patch —

```
@(cd ${GCLVERSION}/h ; \
echo 3 applying EXTRAS patch to h/linux.defs ; \
${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )
@(echo 4 setup ini files for EXTRAS patch ; \
touch ${OBJ}/${SYS}/lib/cfuncs-c.ini ; \
touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

—————

#### — gcl-2.6.7pre.socket.patch —

```
@(cd ${GCLVERSION}/h ; \
echo 3 applying EXTRAS patch to h/linux.defs ; \
${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )
@(echo 4 setup ini files for EXTRAS patch ; \
touch ${OBJ}/${SYS}/lib/cfuncs-c.ini ; \
touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

—————

#### — gcl-2.6.7.socket.patch —

```
@(cd ${GCLVERSION}/h ; \
echo 3 applying EXTRAS patch to h/linux.defs ; \
${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )
@(echo 4 setup ini files for EXTRAS patch ; \
touch ${OBJ}/${SYS}/lib/cfuncs-c.ini ; \
touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

—————

#### — gcl-2.6.8pre.socket.patch —



```
@(cd ${GCLVERSION}/h ; \
  echo 3 applying EXTRAS patch to h/linux.defs ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch ; \
  echo 3a applying EXTRAS patch to h/powerpc-macosx.defs ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.powerpc-macosx.defs.patch )
@(echo 4 setup ini files for EXTRAS patch ; \
  touch ${OBJ}/${SYS}/lib/cfuncs-c.ini ; \
  touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

---

— gcl-2.6.8pre2.socket.patch —

```
@(cd ${GCLVERSION}/h ; \
  echo 3 applying EXTRAS patch to h/linux.defs ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )
@(echo 4 setup ini files for EXTRAS patch ; \
  touch ${OBJ}/${SYS}/lib/cfuncs-c.ini ; \
  touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

---

— gcl-2.6.8pre3.socket.patch —

```
@(cd ${GCLVERSION}/h ; \
  echo 3 applying EXTRAS patch to h/linux.defs ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )
@(echo 4 setup ini files for EXTRAS patch ; \
  touch ${OBJ}/${SYS}/lib/cfuncs-c.ini ; \
  touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

---

— gcl-2.6.8pre4.socket.patch —

```
@(cd ${GCLVERSION}/h ; \
  echo 3 applying EXTRAS patch to h/linux.defs ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )
@(echo 4 setup ini files for EXTRAS patch ; \
  touch ${OBJ}/${SYS}/lib/cfuncs-c.ini ; \
  touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

---

As of this version we no longer use the EXTRAS variable. This has been changed to use the SYSTEM\_OBJS variable, per Camm (Apr 6,2012)

— gcl-2.6.8pre7.h.linux.defs.patch —

```
@(cd ${GCLVERSION}/h ; \
  echo 3 applying EXTRAS patch to h/linux.defs ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )
@(echo 4 setup ini files for EXTRAS patch ; \
  touch ${OBJ}/${SYS}/lib/cfuncs-c.ini ; \
  touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

---

### 7.0.2 read.d patch

The new read-char-no-hang change no longer returns EOF so we have no way to know when the browser is finished talking. This causes AXSERV to hang waiting for more input which never comes. The browser hangs waiting for a response.

— gcl-2.6.8pre3.read.patch —

```
@(cd ${GCLVERSION}/o ; \  
  echo 100 applying read.d patch to o/read.d ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.o.read.d.patch )
```

---

— gcl-2.6.8pre4.read.patch —

```
@(cd ${GCLVERSION}/o ; \  
  echo 100 applying read.d patch to o/read.d ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.o.read.d.patch )
```

---

— gcl-2.6.8pre7.o.read.d.patch —

```
@(cd ${GCLVERSION}/o ; \  
  echo 100 applying read.d patch to o/read.d ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.o.read.d.patch )
```

---

### 7.0.3 fortran patch

Communication over sockets (basically to the NAG fortran library) requires us to have XDR enabled.

— gcl-2.5.2.fortran.patch —

```
@(cd ${GCLVERSION}/h ; \  
  echo 5 applying HAVE_XDR patch to h/linux.h ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.h.patch )
```

---

This patch is no longer necessary as of GCL-2.6.1 because the C preparser symbol HAVE\_XDR is now the default in GCL.

#### 7.0.4 libspad patch

The second patch changes the **unixport/makefile** to reference the **libspad.a** library when building the raw system image. References from **cfuns-c** and **sockio-c** are resolved from **libspad.a**

Additionally this patch contains a temporary fix to the makefile for the windows port. Apparently the `$^` variable is not properly expanded and this works around the problem.

— gcl-2.5.2.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
echo 6 applying libspad.a patch to unixport/makefile ; \  
${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

— gcl-2.6.1.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
echo 6 applying libspad.a patch to unixport/makefile ; \  
${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

— gcl-2.6.2.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
echo 6 applying libspad.a patch to unixport/makefile ; \  
${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

— gcl-2.6.2a.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
echo 6 applying libspad.a patch to unixport/makefile ; \  
${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

— gcl-2.6.3.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
echo 6 applying libspad.a patch to unixport/makefile ; \  
${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

— gcl-2.6.5.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
echo 6 applying libspad.a patch to unixport/makefile ; \  
${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

—————

— gcl-2.6.5w.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
  echo 28 applying libspad.a patch to unixport/makefile ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

—————

In this version of the patch we can delete one of the two changes. The for-loop change has been included in the sources.

— gcl-2.6.6.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
  echo 6 applying libspad.a patch to unixport/makefile ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

—————

— gcl-2.6.7pre.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
  echo 6 applying libspad.a patch to unixport/makefile ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

—————

— gcl-2.6.7.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
  echo 6 applying libspad.a patch to unixport/makefile ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

—————

— gcl-2.6.8pre.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
  echo 6 applying libspad.a patch to unixport/makefile ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

—————

— gcl-2.6.8pre2.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
  echo 6 applying libspad.a patch to unixport/makefile ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

—————

— gcl-2.6.8pre3.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 6 applying libspad.a patch to unixport/makefile ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

---

#### — gcl-2.6.8pre4.libspad.patch —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 6 applying libspad.a patch to unixport/makefile ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

---

#### — gcl-2.6.8pre7.unixport.makefile.patch —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 6 applying libspad.a patch to unixport/makefile ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

---

### 7.0.5 toploop patch

This patch turns off the banner display every time GCL starts. We could use the -batch flag but that would be a pervasive change. It isn't critical to the system builds but we will later be capturing stdin and stdout and we do not want extra information printed.

#### — gcl-2.5.2.toploop.patch —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 7 applying toploop patch to unixport/init_gcl.lsp ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

---

#### — gcl-2.6.1.toploop.patch —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 7 applying toploop patch to unixport/init_gcl.lsp ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

---

#### — gcl-2.6.2.toploop.patch —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 7 applying toploop patch to unixport/init_gcl.lsp ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

---

#### — gcl-2.6.2a.toploop.patch —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 7 applying topleop patch to unixport/init_gcl.lsp ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

—————

— gcl-2.6.3.topleop.patch —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 7 applying topleop patch to unixport/init_gcl.lsp ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

—————

— gcl-2.6.5.topleop.patch —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 7 applying topleop patch to unixport/init_gcl.lsp ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

—————

— gcl-2.6.6.topleop.patch —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 7 applying topleop patch to unixport/init_gcl.lsp ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

—————

— gcl-2.6.7pre.topleop.patch —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 7 applying topleop patch to unixport/init_gcl.lsp ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

—————

— gcl-2.6.7.topleop.patch —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 7 applying topleop patch to unixport/init_gcl.lsp ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

—————

Now, for some reason, lisp needs to tell you what the temporary directory for the compiler will be. We eliminate this noise as well as the banner.

— gcl-2.6.8pre.topleop.patch —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 7 applying topleop patch to unixport/init_gcl.lsp ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

—————

— gcl-2.6.8pre2.toploop.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
  echo 7 applying topleop patch to unixport/init_gcl.lsp ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

—————

— gcl-2.6.8pre3.toploop.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
  echo 7 applying topleop patch to unixport/init_gcl.lsp ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

—————

— gcl-2.6.8pre4.toploop.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
  echo 7 applying topleop patch to unixport/init_gcl.lsp ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

—————

— gcl-2.6.8pre7.unixport.initgcl.lsp.in.patch —

```
@(cd ${GCLVERSION}/unixport ; \  
  echo 7 applying topleop patch to unixport/init_gcl.lsp ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

—————

## 7.0.6 object to float patch

GCL 2.5.2 contains no reference to this function and it was removed. Axiom uses this function so we re-implement it here.

— gcl-2.5.2.objecttofloat.patch —

```
@(cd ${GCLVERSION}/o ; \  
  echo 8 applying object_to_float patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.o.cmpaux.c.patch )
```

—————

This patch is no longer necessary as of GCL-2.6.1 because `object_to_float` is now defined by default.

### 7.0.7 in-package patch

This changes the common lisp 2.0 defined behavior of in-package (throw an error if the package does not exist) so that it mirrors the 1.0 defined behavior (create the package if it does not exist). This patch is intended to be temporary. Axiom's common lisp code should be fixed and this patch removed before shipment.

— gcl-2.5.2.in-package.patch —

```
@(cd ${GCLVERSION}/o ; \  
  echo 9 applying in-package patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.o.package.d.patch )
```

—————  
I believe that all instances of in-package have been fixed and that this patch is no longer necessary.

### 7.0.8 EXIT and MAX\_STACK\_SIZE patches

Axiom uses EXIT as a function. GCL 2.5.2 decided to make it a synonym to the BYE function. We fix this for Axiom. We also patch the MAX\_STACK\_SIZE to be 16Mb.

— gcl-2.5.2.exit.patch —

```
@(cd ${GCLVERSION}/o ; \  
  echo 10 applying EXIT patch ; \  
  echo 18 applying MAX_STACK_SIZE patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.o.main.c.patch )
```

—————  
As of GCL-2.6.1 EXIT is no longer defined.

— gcl-2.6.1.exit.patch —

```
@(cd ${GCLVERSION}/o ; \  
  echo 18 applying MAX_STACK_SIZE patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.o.main.c.patch )
```

—————  
As of GCL-2.6.1 EXIT is no longer defined.

— gcl-2.6.2.exit.patch —

```
@(cd ${GCLVERSION}/o ; \  
  echo 18 applying MAX_STACK_SIZE patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.o.main.c.patch )
```

—————



### 7.0.9 tail-recursive patch

Bill Schelter added tail recursion for Axiom. In order to test it he left code in the system to print a message when the code was executed. We no longer care but it is still in GCL. We patch the call rather than the cmpnote function as cmpnote might have later usage.

Bill Page reported that this tail-recursive patch is no longer necessary for recent releases of GCL. Consequently, it has been disabled for all versions of GCL greater than 2.6.7.

— gcl-2.5.2.tail-recursive.patch —

```
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 11 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.cmpflet.lsp.patch )  
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 12 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.cmpcall.lsp.patch )
```

GCL 2.6.1 renamed the files.

— gcl-2.6.1.tail-recursive.patch —

```
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 11 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpflet.lsp.patch )  
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 12 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpcall.lsp.patch )
```

— gcl-2.6.2.tail-recursive.patch —

```
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 11 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpflet.lsp.patch )  
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 12 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpcall.lsp.patch )
```

— gcl-2.6.2a.tail-recursive.patch —

```
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 11 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpflet.lsp.patch )  
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 12 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpcall.lsp.patch )
```

—————

— gcl-2.6.3.tail-recursive.patch —

```
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 11 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpflet.lsp.patch )  
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 12 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpcall.lsp.patch )
```

—————

— gcl-2.6.5.tail-recursive.patch —

```
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 11 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpflet.lsp.patch )  
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 12 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpcall.lsp.patch )
```

—————

— gcl-2.6.5w.tail-recursive.patch —

```
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 54 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpflet.lsp.patch )  
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 55 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpcall.lsp.patch )
```

—————

— gcl-2.6.6.tail-recursive.patch —

```
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 11 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpflet.lsp.patch )  
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 12 applying tail-recursive noise patch ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.cmpnew.gcl_cmpcall.lsp.patch )
```

—————

## 7.0.10 collectfn fix

GCL-2.6.1 renamed collectfn.lsp to gcl\_collectfn.lsp. We rename it back into place because we have later Makefiles that depend on it. The alternative is to propagate the name changes thru all of those Makefile and they would now need to know about the GCLVERSION variable.

— gcl-2.6.1.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \
  echo 26 copy gcl_collectfn.lsp to collectfn.lsp ; \
  cp gcl_collectfn.lsp ${OBJ}/${SYS}/collectfn.lsp )
```

---

#### — gcl-2.6.2.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \
  echo 26 copy gcl_collectfn.lsp to collectfn.lsp ; \
  cp gcl_collectfn.lsp ${OBJ}/${SYS}/collectfn.lsp )
```

---

In this version we have created a new subdirectory for use by GCL during compile time at obj/sys/lsp. We copy two files from GCL, the collectfn.lsp file and the sys-proclaim.lisp file. The collectfn.lsp contains code which extends the GCL compiler and collects type information. The compile-file function writes this type information to a .fn file as structs. These structs can be loaded and written out as a file of lisp proclaims. The sys-proclaim.lisp file contains the proclaims for GCL's function definitions.

#### — gcl-2.6.2a.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \
  echo 26 copy gcl_collectfn.lsp to ${OBJ}/${SYS}/lsp/collectfn.lsp ; \
  cp gcl_collectfn.lsp ${OBJ}/${SYS}/lsp/collectfn.lsp )
@(cd ${GCLVERSION}/lsp ; \
  echo 27 copy sys-proclaim.lisp to ${OBJ}/${SYS}/lsp/sys-proclaim.lisp ; \
  cp sys-proclaim.lisp ${OBJ}/${SYS}/lsp/sys-proclaim.lisp )
```

---

#### — gcl-2.6.3.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \
  echo 26 copy gcl_collectfn.lsp to ${OBJ}/${SYS}/lsp/collectfn.lsp ; \
  cp gcl_collectfn.lsp ${OBJ}/${SYS}/lsp/collectfn.lsp )
@(cd ${GCLVERSION}/lsp ; \
  echo 27 copy sys-proclaim.lisp to ${OBJ}/${SYS}/lsp/sys-proclaim.lisp ; \
  cp sys-proclaim.lisp ${OBJ}/${SYS}/lsp/sys-proclaim.lisp )
```

---

#### — gcl-2.6.5.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \
  echo 26 copy gcl_collectfn.lsp to ${OBJ}/${SYS}/lsp/collectfn.lsp ; \
  cp gcl_collectfn.lsp ${OBJ}/${SYS}/lsp/collectfn.lsp )
@(cd ${GCLVERSION}/lsp ; \
  echo 27 copy sys-proclaim.lisp to ${OBJ}/${SYS}/lsp/sys-proclaim.lisp ; \
  cp sys-proclaim.lisp ${OBJ}/${SYS}/lsp/sys-proclaim.lisp )
```

—————

— gcl-2.6.5w.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 64 copy gcl_collectfn.lsp to ${OBJ}/${SYS}/lsp/collectfn.lsp ; \  
  cp gcl_collectfn.lsp ${OBJ}/${SYS}/lsp/collectfn.lsp )  
@(cd ${GCLVERSION}/lsp ; \  
  echo 65 copy sys-proclaim.lisp to ${OBJ}/${SYS}/lsp/sys-proclaim.lisp ; \  
  cp sys-proclaim.lisp ${OBJ}/${SYS}/lsp/sys-proclaim.lisp )
```

—————

— gcl-2.6.6.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 26 copy gcl_collectfn.lsp to ${OBJ}/${SYS}/lsp/collectfn.lsp ; \  
  cp gcl_collectfn.lsp ${OBJ}/${SYS}/lsp/collectfn.lsp )  
@(cd ${GCLVERSION}/lsp ; \  
  echo 27 copy sys-proclaim.lisp to ${OBJ}/${SYS}/lsp/sys-proclaim.lisp ; \  
  cp sys-proclaim.lisp ${OBJ}/${SYS}/lsp/sys-proclaim.lisp )
```

—————

— gcl-2.6.7pre.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 26 copy gcl_collectfn.lsp to ${OBJ}/${SYS}/lsp/collectfn.lsp ; \  
  cp gcl_collectfn.lsp ${OBJ}/${SYS}/lsp/collectfn.lsp )  
@(cd ${GCLVERSION}/lsp ; \  
  echo 27 copy sys-proclaim.lisp to ${OBJ}/${SYS}/lsp/sys-proclaim.lisp ; \  
  cp sys-proclaim.lisp ${OBJ}/${SYS}/lsp/sys-proclaim.lisp )
```

—————

— gcl-2.6.7.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 26 copy gcl_collectfn.lsp to ${OBJ}/${SYS}/lsp/collectfn.lsp ; \  
  cp gcl_collectfn.lsp ${OBJ}/${SYS}/lsp/collectfn.lsp )  
@(cd ${GCLVERSION}/lsp ; \  
  echo 27 copy sys-proclaim.lisp to ${OBJ}/${SYS}/lsp/sys-proclaim.lisp ; \  
  cp sys-proclaim.lisp ${OBJ}/${SYS}/lsp/sys-proclaim.lisp )
```

—————

— gcl-2.6.8pre.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \  
  echo 26 copy gcl_collectfn.lsp to ${OBJ}/${SYS}/lsp/collectfn.lsp ; \  
  cp gcl_collectfn.lsp ${OBJ}/${SYS}/lsp/collectfn.lsp )
```

```
@(cd ${GCLVERSION}/lsp ; \
  echo 27 copy sys-proclaim.lisp to ${OBJ}/${SYS}/lsp/sys-proclaim.lisp ; \
  cp sys-proclaim.lisp ${OBJ}/${SYS}/lsp/sys-proclaim.lisp )
```

---

#### — gcl-2.6.8pre2.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \
  echo 26 copy gcl_collectfn.lisp to ${OBJ}/${SYS}/lsp/collectfn.lisp ; \
  cp gcl_collectfn.lisp ${OBJ}/${SYS}/lsp/collectfn.lisp )
@(cd ${GCLVERSION}/lsp ; \
  echo 27 copy sys-proclaim.lisp to ${OBJ}/${SYS}/lsp/sys-proclaim.lisp ; \
  cp sys-proclaim.lisp ${OBJ}/${SYS}/lsp/sys-proclaim.lisp )
```

---

#### — gcl-2.6.8pre3.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \
  echo 26 copy gcl_collectfn.lisp to ${OBJ}/${SYS}/lsp/collectfn.lisp ; \
  cp gcl_collectfn.lisp ${OBJ}/${SYS}/lsp/collectfn.lisp )
@(cd ${GCLVERSION}/lsp ; \
  echo 27 copy sys-proclaim.lisp to ${OBJ}/${SYS}/lsp/sys-proclaim.lisp ; \
  cp sys-proclaim.lisp ${OBJ}/${SYS}/lsp/sys-proclaim.lisp )
```

---

#### — gcl-2.6.8pre4.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \
  echo 26 copy gcl_collectfn.lisp to ${OBJ}/${SYS}/lsp/collectfn.lisp ; \
  cp gcl_collectfn.lisp ${OBJ}/${SYS}/lsp/collectfn.lisp )
@(cd ${GCLVERSION}/lsp ; \
  echo 27 copy sys-proclaim.lisp to ${OBJ}/${SYS}/lsp/sys-proclaim.lisp ; \
  cp sys-proclaim.lisp ${OBJ}/${SYS}/lsp/sys-proclaim.lisp )
```

---

#### — gcl-2.6.8pre7.collectfn.fix —

```
@(cd ${GCLVERSION}/cmpnew ; \
  echo 26 copy gcl_collectfn.lisp to ${OBJ}/${SYS}/lsp/collectfn.lisp ; \
  cp gcl_collectfn.lisp ${OBJ}/${SYS}/lsp/collectfn.lisp )
@(cd ${GCLVERSION}/lsp ; \
  echo 27 copy sys-proclaim.lisp to ${OBJ}/${SYS}/lsp/sys-proclaim.lisp ; \
  cp sys-proclaim.lisp ${OBJ}/${SYS}/lsp/sys-proclaim.lisp )
```

---

## 7.1 The GCL-2.5.2 stanza

### 7.1.1 Configure and Make GCL

We enable several features of GCL. The `-enable-readline` uses GNU readline for the prompts. It has been removed and replaced with `clef` which is Axiom's version of readline. The `-enable-maxpage` is set to allow the image to grow 4 times what it would by default. The `-enable-vssize` allows virtual stack to grow by twice the normal size. The `-enable-statsysbfd` uses a static system bfd library for loading and relocating object files.

Finally we load some routines for performance reasons. `lsp/sys-proclaim` contains common lisp proclaim statements for the various GCL lisp routines. `cmpnew/gcl_collectfn` contains modifications to the common lisp compiler to collect compile-time type information which will be written to `.fn` files as common lisp structs. These `.fn` files can be loaded and turned into common lisp proclaim statements which the compiler can use to generate faster code, mostly fast-path function calls. The call to `compiler::emit-fn` enables the `.fn` file generation whenever `compile-file` is called. We default this code into the image so it is always available.

The `./configure` command takes a few options for building GCL. These need to be changed for various systems so we make these into a variable and move them up to the top level Makefile.

Here we use the newly build `saved_gcl` lisp image to build our lisp image. Next we use this lisp image to compile our tangle function. Next we load the tangle function into a clean lisp image. Finally move the tangle lisp to the named lisp image. This means that all other lisp derived images will know how to tangle.

#### — gclConfigureMake —

```
@(cd ${GCLVERSION} ; \
./configure ${GCLOPTS} ; \
${ENV} ${MAKE} ; \
echo '(progn (load "cmpnew/gcl_collectfn.lisp")' \
        '(load "lsp/sys-proclaim.lisp") (compiler::emit-fn t)' \
        '(system::save-system "${OUT}/lisp"))' | unixport/saved_gcl )
@(cd ${GCLVERSION} ; \
    cp ${BOOKS}/tangle.lisp . ; \
    echo '(compile-file "tangle.lisp")' | ${OUT}/lisp )
@(cd ${GCLVERSION} ; \
    echo '(progn (load "tangle.o")' \
            '(system::save-system "${OUT}/tangle"))' | ${OUT}/lisp )
@mv ${OUT}/tangle ${OUT}/lisp
@echo 9999 lisp tangle version created
```

---

GCL 2.5.2 changes are due to David Mentre (david.mentre@wanadoo.fr). The key problem to solve is that 2.5.2 uses the common lisp 2.0 standard. In Com-

mon Lisp 1.0 if you do (in-package 'foo) and the foo package does not exist it is created. In Common Lisp 2.0 if you do (in-package 'foo) and the foo package does not exist it is an error. The file "gcl-2.5.2/o/package.d" has been changed to keep the old behavior. This is an incorrect fix in the long term. Axiom should be changed everywhere to conform to the common lisp 2.0 standard.

GCL 2.5.2 requires a different Makefile. In particular, GCL 2.5.2 has a different method of building the lisp image. And, just to keep us on our toes, they've renamed the files we need to patch. The new patches are in the zips directory. Patches now are prefixed by GCLVERSION.

When the GCLVERSION variable is "gcl-2.5.2" this stanza will be used. The default version is overwritten. See the top level Makefile.pamphlet.

### — gcl-2.5.2 —

```
# gcl version 2.5.2
OUT=${OBJ}/${SYS}/bin

all:
@echo 1 building ${LSP} ${GCLVERSION}

gcldir:
@echo 2 building ${GCLVERSION}
@tar -xzf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.5.2.fortran.patch}
\getchunk{gcl-2.5.2.objecttofloat.patch}
\getchunk{gcl-2.5.2.in-package.patch}
\getchunk{gcl-2.5.2.exit.patch}
\getchunk{gcl-2.5.2.tail-recursive.patch}
echo '(compiler::link (list (compile-file "${BOOKS}/tangle.lisp")) \
      "${OUT}/lisp" (format nil "(progn (let \
      ((load-path* (cons ~S *load-path*)) (si::load-types* ~S)) \
      (compiler::emit-fn t)) (fmakunbound (quote si::sgc-on)) \
      (when (fboundp (quote si::sgc-on)) (si::sgc-on t)) \
      #-native-reloc (setq compiler::*default-system-p* t))" \
      si::*system-directory* (quote (list \
      #+native-reloc".o" ".lsp")))) "${OBJ}/${SYS}/lib/cfuns-c.o \
      ${OBJ}/${SYS}/lib/sockio-c.o ${OBJ}/${SYS}/lib/libspad.a))' | gcl
@echo 13 finished system build on 'date' | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )
```

```

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/cc1
@ ( cd cc1 ; ${ENV} ${MAKE} document )

clean:
@echo 17 cleaning ${LSP}/cc1
@ ( cd cc1 ; ${ENV} ${MAKE} clean )

```

## 7.2 The GCL-2.6.1 stanza

GCL 2.6.1 has support for `-enable-static` for static linking. Axiom will eventually support static images.

GCL 2.6.1 is supposed to build and run on Windows under mingw. Axiom will eventually support a windows version.

GCL 2.6.1 continues its tradition of renaming files we need to patch. The new patches are in the zips directory.

When the GCLVERSION variable is “gcl-2.6.1” this stanza will be used. It will overwrite the default version. See the top level Makefile.pamphlet.

### — gcl-2.6.1 —

```

# gcl version 2.6.1
OUT=${OBJ}/${SYS}/bin

all:
@echo 1 building ${LSP} ${GCLVERSION}

gcldir:
@echo 2 building ${GCLVERSION}
@tar -zxf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.6.1.socket.patch}
\getchunk{gcl-2.6.1.libspad.patch}
\getchunk{gcl-2.6.1.toploop.patch}
\getchunk{gcl-2.6.1.exit.patch}
\getchunk{gcl-2.6.1.tail-recursive.patch}
\getchunk{gcl-2.6.1.collectfn.fix}
\getchunk{gclConfigureMake}
@echo 13 finished system build on `date` | tee >gcldir

ccldir: ${LSP}/cc1/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/cc1
@mkdir -p ${OBJ}/${SYS}/cc1
@ ( cd cc1 ; ${ENV} ${MAKE} )

```



```

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@ ( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@ ( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 17 cleaning ${LSP}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} clean )

```

### 7.3 The GCL-2.6.2 stanza

GCL 2.6.2 has support for `-enable-static` for static linking. Axiom will eventually support static images.

GCL 2.6.2 is supposed to build and run on Windows under mingw. Axiom will eventually support a windows version.

GCL 2.6.2 continues its tradition of renaming files we need to patch. The new patches are in the zips directory.

When the GCLVERSION variable is “gcl-2.6.2” this stanza will be written. It will overwrite the default version. See the top level Makefile.pamphlet.

#### — gcl-2.6.2 —

```

# gcl version 2.6.2
OUT=${OBJ}/${SYS}/bin

all:
@echo 1 building ${LSP} ${GCLVERSION}

gclmdir:
@echo 2 building ${GCLVERSION}
@tar -zxf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.6.2.socket.patch}
\getchunk{gcl-2.6.2.libspad.patch}
\getchunk{gcl-2.6.2.toploop.patch}
\getchunk{gcl-2.6.2.exit.patch}
\getchunk{gcl-2.6.2.tail-recursive.patch}
\getchunk{gcl-2.6.2.collectfn.fix}
\getchunk{gclConfigureMake}
@echo 13 finished system build on 'date' | tee >gclmdir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL

```

```

@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@ ( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@ ( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 17 cleaning ${LSP}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} clean )

```

---

## 7.4 Directory move

The latest version of GCL no longer uses the version number in the directory name. We fix that here.

— **gcl-2.6.2a-mvdir** —

```

@echo 27 renaming gcl to ${GCLVERSION}
@mv gcl gcl-2.6.2a

```

---

## 7.5 The GCL-2.6.2a stanza

GCL-2.6.2a has a variable `si::*optimize-maximum-pages*` which defaults to `t` which eliminates an old problem of spinning the gc for temporary allocations of eg. strings in a 32 page space on a heap growing to quite large size. You might want to investigate (time ..) and (room) on computationally intensive jobs with and without this variable set.

This stanza will be written when the GCLVERSION variable is “gcl-2.6.2a”. It will overwrite the default version. See the top level Makefile.pamphlet.

— **gcl-2.6.2a** —

```

# gcl version 2.6.2a
OUT=${OBJ}/${SYS}/bin

all:
@echo 1 building ${LSP} ${GCLVERSION}

gcldir:
@echo 2 building ${GCLVERSION}

```

```

@tar -zxf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.6.2a-mvdir}
\getchunk{gcl-2.6.2a.socket.patch}
\getchunk{gcl-2.6.2a.libspad.patch}
\getchunk{gcl-2.6.2a.toploop.patch}
\getchunk{gcl-2.6.2a.tail-recursive.patch}
\getchunk{gcl-2.6.2a.collectfn.fix}
\getchunk{gclConfigureMake}
@echo 13 finished system build on 'date' | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@ ( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@ ( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 17 cleaning ${LSP}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} clean )

```

---

## 7.6 Directory move

The latest version of GCL no longer uses the version number in the directory name. We fix that here.

— **gcl-2.6.3-mvdir** —

```

@echo 27 renaming gcl to ${GCLVERSION}
@mv gcl gcl-2.6.3

```

---

## 7.7 The GCL-2.6.3 stanza

This stanza will be written when the GCLVERSION variable is “gcl-2.6.3”. It will overwrite the default version. See the top level Makefile.pamphlet.

— **gcl-2.6.3** —

```

# gcl version 2.6.3

```

```

OUT=${OBJ}/${SYS}/bin

all:
@echo 1 building ${LSP} ${GCLVERSION}

gcldir:
@echo 2 building ${GCLVERSION}
@tar -zxf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.6.3-mvdir}
\getchunk{gcl-2.6.3-socket.patch}
\getchunk{gcl-2.6.3-libspad.patch}
\getchunk{gcl-2.6.3-toploop.patch}
\getchunk{gcl-2.6.3-tail-recursive.patch}
\getchunk{gcl-2.6.3-collectfn.fix}
\getchunk{gclConfigureMake}
@echo 13 finished system build on 'date' | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 17 cleaning ${LSP}/ccl
@( cd ccl ; ${ENV} ${MAKE} clean )

```

---

## 7.8 The GCL-2.6.5 stanza

This stanza will be written when the GCLVERSION variable is “gcl-2.6.5”. It will overwrite the default version. See the top level Makefile.pamphlet.

— gcl-2.6.5 —

```

# gcl version 2.6.5
OUT=${OBJ}/${SYS}/bin

all:
@echo 1 building ${LSP} ${GCLVERSION}

```

```

gcldir:
@echo 2 building ${GCLVERSION}
@tar -zxf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.6.5.socket.patch}
\getchunk{gcl-2.6.5.libspad.patch}
\getchunk{gcl-2.6.5.toploop.patch}
\getchunk{gcl-2.6.5.h.gmp\_wrappers.h.patch}
\getchunk{gcl-2.6.5.tail-recursive.patch}
\getchunk{gcl-2.6.5.collectfn.fix}
\getchunk{gclConfigureMake}
@echo 13 finished system build on `date` | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@ ( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@ ( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 17 cleaning ${LSP}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} clean )

```

---

## 7.9 The GCL-2.6.5w stanza

This stanza will be written when the GCLVERSION variable is “gcl-2.6.5w”. It will overwrite the default version. See the top level Makefile.pamphlet.

This is version for Windows. We have moved forward several patches from the GCL standard but are not ready to accept the latest GCL version because it breaks Axiom. We only include the patches that are needed for Windows. We untar the gcl-2.6.5 version and then rename it to gcl-2.6.5w

— gcl-2.6.5w —

```

# gcl version 2.6.5w
OUT=${OBJ}/${SYS}/bin

all:
@echo 110 building ${LSP} ${GCLVERSION}

```

```

gcldir:
@echo 111 building ${GCLVERSION}
@tar -zxvf ${ZIPS}/gcl-2.6.5.tgz
@mv gcl-2.6.5 ${GCLVERSION}
\getchunk{gcl-2.6.5w.socket.patch}
\getchunk{gcl-2.6.5w.libspad.patch}
\getchunk{gcl-2.6.5w.toploop.patch}
\getchunk{gcl-2.6.5w.h.gmp\_wrappers.h.patch}
\getchunk{gcl-2.6.5w.tail-recursive.patch}
\getchunk{gcl-2.6.5w.collectfn.fix}
\getchunk{gcl-2.6.5w.h.mingw.defs.patch}
\getchunk{gcl-2.6.5w.o.alloc.c.patch}
\getchunk{gcl-2.6.5w.o.mingfile.c.patch}
\getchunk{gcl-2.6.5w.o.unixfsys.c.patch}
\getchunk{gclConfigureMake}
@echo 112 finished system build on 'date' | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 113 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 114 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@ ( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 115 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@ ( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 116 cleaning ${LSP}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} clean )

```

---

## 7.10 The GCL-2.6.6 stanza

This stanza will be written when the GCLVERSION variable is “gcl-2.6.6”. It will overwrite the default version. See the top level Makefile.pamphlet.

— gcl-2.6.6 —

```

# gcl version 2.6.6
OUT=${OBJ}/${SYS}/bin

```

```

all:

```

```

@echo 1 building ${LSP} ${GCLVERSION}

gcldir:
@echo 2 building ${GCLVERSION}
@tar -zxvf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.6.6.h.linux.h.patch}
\getchunk{gcl-2.6.6.socket.patch}
\getchunk{gcl-2.6.6.libspad.patch}
\getchunk{gcl-2.6.6.toploop.patch}
\getchunk{gcl-2.6.6.tail-recursive.patch}
\getchunk{gcl-2.6.6.collectfn.fix}
\getchunk{gclConfigureMake}
@echo 13 finished system build on `date` | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 17 cleaning ${LSP}/ccl
@( cd ccl ; ${ENV} ${MAKE} clean )

```

---

## 7.11 The GCL-2.6.7pre stanza

This stanza will be written when the GCLVERSION variable is “gcl-2.6.7pre”. It will overwrite the default version. See the top level Makefile.pamphlet.

— gcl-2.6.7pre —

```

# gcl version 2.6.7pre
OUT=${OBJ}/${SYS}/bin

all:
@echo 1 building ${LSP} ${GCLVERSION}

gcldir:
@echo 2 building ${GCLVERSION}

```

```

@tar -zxf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.6.7pre.socket.patch}
\getchunk{gcl-2.6.7pre.libspad.patch}
\getchunk{gcl-2.6.7pre.toploop.patch}
\getchunk{gcl-2.6.7pre.collectfn.fix}
\getchunk{gclConfigureMake}
@echo 13 finished system build on 'date' | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@ ( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@ ( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 17 cleaning ${LSP}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} clean )

```

—

## 7.12 The GCL-2.6.7 stanza

This stanza will be written when the GCLVERSION variable is “gcl-2.6.7”. It will overwrite the default version. See the top level Makefile.pamphlet.

— gcl-2.6.7 —

```

# gcl version 2.6.7
OUT=${OBJ}/${SYS}/bin

all:
@echo 1 building ${LSP} ${GCLVERSION}

gcldir:
@echo 2 building ${GCLVERSION}
@tar -zxf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.6.7.configure.in.patch}
\getchunk{gcl-2.6.7.socket.patch}
\getchunk{gcl-2.6.7.libspad.patch}
\getchunk{gcl-2.6.7.toploop.patch}
\getchunk{gcl-2.6.7.collectfn.fix}

```



```

\getchunk{gclConfigureMake}
@echo 13 finished system build on `date` | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@(` cd ccl ; ${ENV} ${MAKE} `)

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@(` cd ccl ; ${DOCUMENT} ${NOISE} Makefile `)

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@(` cd ccl ; ${ENV} ${MAKE} document `)

clean:
@echo 17 cleaning ${LSP}/ccl
@(` cd ccl ; ${ENV} ${MAKE} clean `)

```

---

### 7.13 The GCL-2.6.8pre stanza

This stanza will be written when the GCLVERSION variable is “gcl-2.6.8pre”. It will overwrite the default version. See the top level Makefile.pamphlet.

— gcl-2.6.8pre —

```

# gcl version 2.6.8pre
OUT=${OBJ}/${SYS}/bin

all:
@echo 1 building ${LSP} ${GCLVERSION}

gcldir:
@echo 2 building ${GCLVERSION}
@tar -zxf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.6.8pre.socket.patch}
\getchunk{gcl-2.6.8pre.libspad.patch}
\getchunk{gcl-2.6.8pre.toploop.patch}
\getchunk{gcl-2.6.8pre.collectfn.fix}
\getchunk{gclConfigureMake}
@echo 13 finished system build on `date` | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/ccl

```

```

@mkdir -p ${OBJ}/${SYS}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@ ( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@ ( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 17 cleaning ${LSP}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} clean )

```

---

## 7.14 The GCL-2.6.8pre2 stanza

This stanza will be written when the GCLVERSION variable is “gcl-2.6.8pre2”. It will overwrite the default version. See the top level Makefile.pamphlet.

— gcl-2.6.8pre2 —

```

# gcl version 2.6.8pre2
OUT=${OBJ}/${SYS}/bin

all:
@echo 1 building ${LSP} ${GCLVERSION}

gcldir:
@echo 2 building ${GCLVERSION}
@tar -zxf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.6.8pre2.socket.patch}
\getchunk{gcl-2.6.8pre2.libspad.patch}
\getchunk{gcl-2.6.8pre2.toploop.patch}
\getchunk{gcl-2.6.8pre2.collectfn.fix}
\getchunk{gclConfigureMake}
@echo 13 finished system build on ‘date’ | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@ ( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

```

```

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@ ( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 17 cleaning ${LSP}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} clean )

```

## 7.15 The GCL-2.6.8pre3 stanza

This stanza will be written when the GCLVERSION variable is “gcl-2.6.8pre3”. It will overwrite the default version. See the top level Makefile.pamphlet.

— gcl-2.6.8pre3 —

```

# gcl version 2.6.8pre3
OUT=${OBJ}/${SYS}/bin

all:
@echo 1 building ${LSP} ${GCLVERSION}

gcldir:
@echo 2 building ${GCLVERSION}
@tar -zxf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.6.8pre3.socket.patch}
\getchunk{gcl-2.6.8pre3.libspad.patch}
\getchunk{gcl-2.6.8pre3.toploop.patch}
\getchunk{gcl-2.6.8pre3.collectfn.fix}
\getchunk{gcl-2.6.8pre3.read.patch}
\getchunk{gclConfigureMake}
@echo 13 finished system build on `date` | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@ ( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@ ( cd ccl ; ${ENV} ${MAKE} document )

```

```

clean:
@echo 17 cleaning ${LSP}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} clean )

```

## 7.16 The GCL-2.6.8pre4 stanza

This stanza will be written when the GCLVERSION variable is “gcl-2.6.8pre4”. It will overwrite the default version. See the top level Makefile.pamphlet.

— gcl-2.6.8pre4 —

```

# gcl version 2.6.8pre4
OUT=${OBJ}/${SYS}/bin

all:
@echo 1 building ${LSP} ${GCLVERSION}

gcldir:
@echo 2 building ${GCLVERSION}
@tar -zxf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.6.8pre4.socket.patch}
\getchunk{gcl-2.6.8pre4.libspad.patch}
\getchunk{gcl-2.6.8pre4.toploop.patch}
\getchunk{gcl-2.6.8pre4.collectfn.fix}
\getchunk{gcl-2.6.8pre4.read.patch}
\getchunk{gclConfigureMake}
@echo 13 finished system build on `date` | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@ ( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@ ( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 17 cleaning ${LSP}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} clean )

```

---

## 7.17 The GCL-2.6.8pre7 stanza

This stanza will be written when the GCLVERSION variable is “gcl-2.6.8pre7”. It will overwrite the default version. See the top level Makefile.pamphlet.

— gcl-2.6.8pre7 —

```
# gcl version 2.6.8pre7
OUT=${OBJ}/${SYS}/bin

all:
@echo 1 building ${LSP} ${GCLVERSION}

gcldir:
@echo 2 building ${GCLVERSION}
@tar -zxvf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.6.8pre7.h.linux.defs.patch}
\getchunk{gcl-2.6.8pre7.o.read.d.patch}
\getchunk{gcl-2.6.8pre7.unixport.initgcl.lsp.in.patch}
\getchunk{gcl-2.6.8pre7.unixport.makefile.patch}
\getchunk{gcl-2.6.8pre7.collectfn.fix}
\getchunk{gclConfigureMake}
@echo 13 finished system build on 'date' | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@{ cd ccl ; ${ENV} ${MAKE} }

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@{ cd ccl ; ${DOCUMENT} ${NOISE} Makefile }

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@{ cd ccl ; ${ENV} ${MAKE} document }

clean:
@echo 17 cleaning ${LSP}/ccl
@{ cd ccl ; ${ENV} ${MAKE} clean }
```

---

## 8 Gnu Common Lisp 2.5

GCL 2.5 requires a different Makefile. In particular, GCL 2.5 has a different method of building the lisp image. And, just to keep us on our toes, they've renamed the files we need to patch. The new patches are in the zips directory. Patches now are prefixed by GCLVERSION.

### 8.0.1 socket patch

This patch is to **h/386-linux.defs** to include two global variables, **EXTRAS** and **EXTRA\_LIB**

```
EXTRAS = ${OBJ}/${SYS}/lib/cfuncs-c.o ${OBJ}/${SYS}/lib/sockio-c.o
EXTRA_LIB=${OBJ}/${SYS}/lib/libspad.a
```

The **EXTRAS** variable is used to include two files into the running image. The **cfuncs-c** file contains low-level directory manipulation routines. The **sockio-c** file contains low level socket code. Note that versions of GCL beyond gcl-2.4.1 may have routines already available. If so the Axiom references to these routines should be rewritten.

We also need to create two .ini files, one for cfuncs-c and one for sockio-c. These are referenced in the gcl-2.4.1 makefiles but, since no initialization is needed, we simply create empty files.

— **gcl-2.5.socket.patch** —

```
@(cd ${GCLVERSION}/h ; \
  echo 3 applying EXTRAS patch to h/linux.defs ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.defs.patch )
@(echo 4 setup ini files for EXTRAS patch ; \
  touch ${OBJ}/${SYS}/lib/cfuncs-c.ini ; \
  touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

---

### 8.0.2 fortran patch

Communication over sockets (basically to the NAG fortran library) requires us to have XDR enabled.

— **gcl-2.5.fortran.patch** —

```
@(cd ${GCLVERSION}/h ; \
  echo 5 applying HAVE_XDR patch to h/linux.h ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.linux.h.patch )
```

---

### 8.0.3 libspad patch

The second patch changes the **unixport/makefile** to reference the **libspad.a** library when building the raw system image. References from **cfuns-c** and **sockio-c** are resolved from **libspad.a**

— **gcl-2.5.libspad.patch** —

```
@(cd ${GCLVERSION}/unixport ; \  
  echo 6 applying libspad.a patch to unixport/makefile ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

---

### 8.0.4 toploop patch

This patch turns off the banner display every time GCL starts. We could use the **-batch** flag but that would be a pervasive change. It isn't critical to the system builds but we will later be capturing stdin and stdout and we do not want extra information printed.

— **gcl-2.5.toploop.patch** —

```
@(cd ${GCLVERSION}/unixport ; \  
  echo 7 applying toploop patch to unixport/init_gcl.lsp ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.in.patch )
```

---

### 8.0.5 xdrfuns bug patch

We enabled XDR only to find out that there is a missing argument in the call to **FEError** from **xdrstdio\_create**. We fix this here and send a patch to the GCL people. It should be fixed in future releases and this patch should die.

— **gcl-2.5.xdrfuns.patch** —

```
@(cd ${GCLVERSION}/o ; \  
  echo 8 applying XDR patch to o/xdrfuns.c ; \  
  ${PATCH} <${SPD}/zips/${GCLVERSION}.o.xdrfuns.c.patch )
```

---

## 8.1 The GCL-2.5 stanza

This stanza will be written when the **GCLVERSION** variable is “gcl-2.5”. It will overwrite the default version. See the top level **Makefile.pamphlet**.

— **gcl-2.5** —

```
# gcl version 2.5  
OUT=${OBJ}/${SYS}/bin  
  
all:  
@echo 1 building ${LSP} ${GCLVERSION}
```

```

gclmdir:
@echo 2 building ${GCLVERSION}
@tar -zxvf ${ZIPS}/${GCLVERSION}.tgz
@mv gcl ${GCLVERSION}
\getchunk{gcl-2.5.socket.patch}
\getchunk{gcl-2.5.fortran.patch}
\getchunk{gcl-2.5.libspad.patch}
\getchunk{gcl-2.5.toploop.patch}
\getchunk{gcl-2.5.xdrfuns.patch}
@(cd ${GCLVERSION} ; \
./configure --enable-vssize=65536 ; \
${ENV} ${MAKE} ; \
cp unixport/saved_gcl ${OUT}/lisp )
@echo 9 finished system build on 'date' | tee >gclmdir

ccldir: ${LSP}/ccl/Makefile
@echo 10 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 11 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@ ( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 12 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@ ( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 13 cleaning ${LSP}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} clean )

```

---

## 9 Gnu Common Lisp 2.4.1 (The default build)

### 9.0.1 socket patch

This patch is to **h/386-linux.defs** to include two global variables, **EXTRAS** and **EXTRA\_LIB**

```

EXTRAS = ${OBJ}/${SYS}/lib/cfuns-c.o ${OBJ}/${SYS}/lib/sockio-c.o
EXTRA_LIB=${OBJ}/${SYS}/lib/libspad.a

```

The **EXTRAS** variable is used to include two files into the running image. The **cfuns-c** file contains low-level directory manipulation routines. The **sockio-c**



file contains low level socket code. Note that versions of GCL beyond gcl-2.4.1 may have routines already available. If so the Axiom references to these routines should be rewritten.

We also need to create two .ini files, one for cfuns-c and one for sockio-c. These are referenced in the gcl-2.4.1 makefiles but, since no initialization is needed, we simply create empty files.

— **gcl-2.4.1.socket.patch** —

```
@(cd ${GCLVERSION}/h ; \
  echo 16 applying EXTRAS patch to h/386-linux.defs ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.386-linux.defs.patch )
@(echo 17 setup ini files for EXTRAS patch ; \
  touch ${OBJ}/${SYS}/lib/cfuns-c.ini ; \
  touch ${OBJ}/${SYS}/lib/sockio-c.ini )
```

---

### 9.0.2 fortran patch

Communication over sockets (basically to the NAG fortran library) requires us to have XDR enabled.

— **gcl-2.4.1.fortran.patch** —

```
@(cd ${GCLVERSION}/h ; \
  echo 18 applying HAVE_XDR patch to h/386-linux.h ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.h.386-linux.h.patch )
```

---

### 9.0.3 libspad patch

The second patch changes the **unixport/makefile** to reference the **libspad.a** library when building the raw system image. References from **cfuns-c** and **sockio-c** are resolved from **libspad.a**

— **gcl-2.4.1.libspad.patch** —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 19 applying libspad.a patch to unixport/makefile ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.makefile.patch )
```

---

### 9.0.4 toploop patch

This patch turns off the banner display every time GCL starts. We could use the -batch flag but that would be a pervasive change. It isn't critical to the system builds but we will later be capturing stdin and stdout and we do not want extra information printed.

— **gcl-2.4.1.toploop.patch** —

```
@(cd ${GCLVERSION}/unixport ; \
  echo 20 applying topleop patch to unixport/init_gcl.lsp ; \
  ${PATCH} <${SPD}/zips/${GCLVERSION}.unixport.init_gcl.lsp.patch )
```

---

## 10 The Makefile

### 10.1 GCL already installed

On some systems, notably freebsd, we assume that GCL is already installed and available using the command gcl. In that case we need to extract this Makefile instead of the standard one.

— gcl-system —

```
# locally installed GCL
OUT=${OBJ}/${SYS}/bin

all:
@echo 21 building ${LSP} ${GCLVERSION}

gcldir:
@echo 22 building for ${GCLVERSION}
echo '(compiler::link nil "${OUT}/lisp" \
  (format nil \
    "(progn \
      (let ((*load-path* (cons ~S *load-path*))) (si::*load-types* ~S)) \
        (compiler::emit-fn t)) \
        (when (fboundp (quote si::sgc-on)) (si::sgc-on t)) \
        (setq compiler::*default-system-p* t))" \
      si::*system-directory* (quote (list ".lsp")))) \
  "${OBJ}/${SYS}/lib/cfun-c.o \
  ${OBJ}/${SYS}/lib/sockio-c.o \
  ${OBJ}/${SYS}/lib/libspad.a")' | gcl
@echo 23 finished gcl build on 'date' | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 21 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 22 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 23 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
```

```

@ ( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 24 cleaning ${LSP}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} clean )

_____

__ * __

OUT=${OBJ}/${SYS}/bin

all:
@echo 14 building ${LSP} ${GCLVERSION}

gcldir:
@echo 15 building ${GCLVERSION}
@tar -zxf ${ZIPS}/${GCLVERSION}.tgz
\getchunk{gcl-2.4.1.socket.patch}
\getchunk{gcl-2.4.1.fortran.patch}
\getchunk{gcl-2.4.1.libspad.patch}
\getchunk{gcl-2.4.1.toploop.patch}
@ ( cd ${GCLVERSION} ; \
./configure --enable-vssize=65536 ; \
${ENV} ${MAKE} ; \
cp unixport/saved_gcl ${OUT}/lisp )
@echo 21 finished system build on 'date' | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 22 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 23 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@ ( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 24 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@ ( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 25 cleaning ${LSP}/ccl
@ ( cd ccl ; ${ENV} ${MAKE} clean )

_____

```

## 10.2 The GCL-cygwin stanza

This stanza will be written when the GCLVERSION variable is “gcl-cygwin”. It will overwrite the default version. See the top level Makefile.pamphlet.

The compiler::link function call was suggested by Camm as a way around patching the lisp system. The function creates a new lisp image ld and C objects prior to save-system.

```
echo '(compiler::link \
```

The first argument can be used to compile and load lisp code.

```
(list (compile-file "${BOOKS}/tangle.lisp")) \
```

The second argument gives the file location for the save-system

```
"${OUT}/lisp" \
```

The third argument sets up various internal variables, including the

```
(format nil \
"(progn \
  (let ((*load-path* (cons ~S *load-path*)) \
    (si::*load-types* ~S)) \
    (compiler::emit-fn t)) \
  (setq *tmp-dir* "${TMP}") \
  (makunbound (quote *system-banner*)) \
  (when (fboundp (quote si::sgc-on)) (si::sgc-on t)) \
  #-native-reloc (setq compiler::*default-system-p* t))" \
si::*system-directory* \
(quote (list #+native-reloc ".o" ".lisp")))) \
```

The fourth argument is a list of binaries to link

```
"${OBJ}/${SYS}/lib/cfuns-c.o \
${OBJ}/${SYS}/lib/sockio-c.o \
${OBJ}/${SYS}/lib/libspad.a)"' \
```

and the compiler::link command is piped into the lisp image.

```
| unixport/saved_gcl )
```

— gcl-cygwin —

```
# gcl version cygwin
OUT=${OBJ}/${SYS}/bin
```

```
all:
@echo 1 building ${LSP} ${GCLVERSION}
```

```
gcldir:
@echo 2 building ${GCLVERSION}
```

```

@tar -zxvf ${ZIPS}/${GCLVERSION}.tgz
(cd ${GCLVERSION} ; \
./configure ${GCLOPTS} ; \
${ENV} ${MAKE} ; \
echo '(compiler::link (list (compile-file "${BOOKS}/tangle.lisp")) "${OUT}/lisp" (format nil "(progn (
@echo 13 finished system build on `date` | tee >gcldir

ccldir: ${LSP}/ccl/Makefile
@echo 14 building CCL
@mkdir -p ${INT}/ccl
@mkdir -p ${OBJ}/${SYS}/ccl
@( cd ccl ; ${ENV} ${MAKE} )

${LSP}/ccl/Makefile: ${LSP}/ccl/Makefile.pamphlet
@echo 15 making ${LSP}/ccl/Makefile from ${LSP}/ccl/Makefile.pamphlet
@( cd ccl ; ${DOCUMENT} ${NOISE} Makefile )

document:
@echo 16 making docs in ${LSP}
@mkdir -p ${INT}/doc/lsp/ccl
@( cd ccl ; ${ENV} ${MAKE} document )

clean:
@echo 17 cleaning ${LSP}/ccl
@( cd ccl ; ${ENV} ${MAKE} clean )

```

---

## References

- [1] nothing