Miscellaneous FAQ

This chapter summarizes important concepts found throughout this manual in the form of frequently asked questions and their answers.

Q. Why can't I link my 32-bit application with a 64-bit library?

A. There are differences in the subroutine calling interface.

Q. How Do I disable KAP ?

A. To disable KAP add the following to your compile line
-WK,-o=0,-ro=0,-so=0

Q. How can I see what KAP does to my code?

A. To view the file add the following to your compile line
-WK,-cmp=file.cmp

Q. My */tmp* directory becomes full when I do a compile.
What should I do ?

A. %**`setenv TMP_DIR /directory`** on a free partition.

Q. How do I know which compiler is being invoked ?

A. *cc -show* will show each of the components as they are being run.

Q. My 64-bit shared application gets mysterious messages from *rld* at runtime.What is the problem ?

A. It's possible that you are linking with 32-bit *.so's*. Check and reset your *_RLD_ROOT* environment variable.

Q. How can I avoid always setting -**32** or -**64** on my compilation command line ?

A. *%setenv SGI_ABI -32* or
%setenv SGI_ABI -64.

Q. How do I know if my code software pipelined OK ?

A. Compile your file (say *foo.f*)*-64 -O3 -mips4 -S*. Then *grep* #<*swp foo.s*#
#<swpf will be printed for loops that failed to software pipeline.
#<swps will be printed for statistics and other info about the loops that did software pipeline.

Q. I compiled -**O**, but my code did not software pipeline. Why not?

A. Software pipelining occurs at -**O3**. -**O** implies -**O2**.

Q. Ok, I now compiled -**O3**. Why didn't my code software pipeline?

A. Does it have a call or branch in the inner loop?
Is the inner loop too long?
Does the inner loop execute too few iterations?
Are there divide recurrences in your inner loop?

Q. What predefine should my source code look at to know if its being compiled 32-bit or 64-bit.

A. One way is:
#if (_MIPS_SZLONG == 64)

Q. How can I force a large loop to be unrolled by the compiler?

A Add -**OPT:unroll_size=1000** to your command line?

Q. Why does my application gets different floating point numbers in 64-bit mode?

A. If its compiled with optimization, the order of operations may different resulting in slightly different calculations due to rounding. *madd* instructions also round differently than a multiply and an add. The math libraries use different algorithms.

Q. Why doesn't 64-bit **printf** print my floating point values?

A. C Programs using **printf()** to print out floating point (or double) values must include stdio.h.

```
%cat example.c
main()
{
 float f = 3.14;
 double d = 2.718;
 printf ("%f\n",f);
 printf ("%e\n",d);
}
```

```
%cc -32 example.c
%a.out
3.140000
2.718000e+00
```

```
%cc -64 example.c
%a.out
0.000000
1.302121e-315
```

```
%cat example_fix.c
#include <stdio.h>
main()
{
float f = 3.14;
double d = 2.718;
printf ("%f\n",f);
printf ("%e\n",d);
}

% cc -64 example_fix.c
%a.out
3.140000
2.718000e+00
```