

## Re: Thread local storage not working with -fPIC and shared objects

---

*Source:* <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/hackers/2007-03/msg00258.html>

---

- *From:* Garrett Cooper <[youshi10@xxxxxxxxxxxxxxxxxxxx](mailto:youshi10@xxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Fri, 30 Mar 2007 20:14:39 -0800
- 

Pieter de Goeje wrote:

Hello List,

I have these files:

```
--- loader.cpp ---  
#include <stdio.h>  
#include "tls.h"
```

```
int main() {  
    tls = 0;  
    printf("%d\n", tls);  
}
```

```
--- tls.cpp ---  
#include "tls.h"  
int __thread tls;
```

```
--- tls.h ---  
extern __thread int tls;
```

When I compile them like this:

```
c++ -fPIC -o tls.so tls.cpp -shared  
c++ -fPIC -o loader loader.cpp tls.so
```

And run the resulting program, I get:

```
pyotr@nox:~/projects/misc/tls> ./loader  
/libexec/ld-elf.so.1: ./loader: Unsupported relocation type 37 in non-PLT relocations
```

When I omit -fPIC, it runs fine. But I need fPIC for the shared object on amd64 arch. I've tried it on Linux/i386 (gcc 4.1) and it ran fine (with fPIC).

Much to my surprise however, a particularly large application I'm working on did compile & run on FreeBSD/amd64 using -fpic (lowercase) and gcc 4.3. Trying -fpic on FreeBSD/i386 resulted in failure.

FYI, I need tls to work because I'm using OpenMP's tls (#pragma omp threadprivate()) support in gcc 4.3.

Re: Thread local storage not working with -fPIC and shared objects

The workaround I found on FreeBSD/amd64 was linking the main executable with -fno-PIC, or building everything with -fpic. (both workarounds didn't work on FreeBSD/i386)

I would be grateful if someone could shed some light on this.

Regards,  
Pieter de Goeje

Pieter,

Did you know that -fPIC and -fpic aren't the same? From

<<http://gcc.gnu.org/onlinedocs/gcc-4.1.2/gcc/Code-Gen-Options.html#Code-Gen-Options>>:

| -fpic |

Generate position-independent code (PIC) suitable for use in a shared library, if supported for the target machine. Such code accesses all constant addresses through a global offset table (GOT). The dynamic loader resolves the GOT entries when the program starts (the dynamic loader is not part of GCC; it is part of the operating system). If the GOT size for the linked executable exceeds a machine-specific maximum size, you get an error message from the linker indicating that -fpic does not work; in that case, recompile with -fPIC instead. (These maximums are 8k on the SPARC and 32k on the m68k and RS/6000. The 386 has no such limit.)

Position-independent code requires special support, and therefore works only on certain machines. For the 386, GCC supports PIC for System V but not for the Sun 386i. Code generated for the IBM RS/6000 is always position-independent.

| -fPIC |

If supported for the target machine, emit position-independent code, suitable for dynamic linking and avoiding any limit on the size of the global offset table. This option makes a difference on the m68k, PowerPC and SPARC.

Position-independent code requires special support, and therefore works only on certain machines.

Cheers,  
-Garrett

---

freebsd-hackers@xxxxxxxxxxx mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-hackers>

To unsubscribe, send any mail to "freebsd-hackers-unsubscribe@xxxxxxxxxxx"