picolibc

A C Library for Smaller Systems

Keith Packard
Principal Engineer
SiFive
keith.packard@sifive.com

Embedded Libc Needs

- Math Functions
  - Often for soft-float processors
- String Functions
  - Ideally accelerated for architecture
- Stdio
  - Largely for debugging

Small System Constraints

- Small Memory
  - RAM is more constrained than ROM
- No heap
  - malloc can easily fail
- Limited floating point
  - May have only 32-bit floats
  - May have none at all

Current 32-bit Libc Options

- newlib and newlib-nano
  - Designed for systems with an OS
  - libgloss wraps OS functions for newlib
  - stdio is fast, but malloc-intensive
- various proprietary options
  - closed source
  - unable to fix

“Fixing” newlib

- Replace stdio
  - Must not malloc
  - Should use as little RAM as possible
  - Retain full C semantics
- Discard libgloss
  - No value here for bare-metal systems

picolibc

- newlib math, i18n, strings
  - good performance, wide support
- stdio adapted from AVR libc
  - FILE takes just 20 bytes of RAM

stdio

```
struct __file {
    unsigned char unget;                 /* ungetc() buffer */
    uint8_t  flags;                      /* flags, see below */
    int  len;                            /* characters read or written so far */
    int  (*put)(char, struct __file *);  /* function to write one char to device */
    int  (*get)(struct __file *);        /* function to read one char from device */
    int  (*flush)(struct __file *);      /* function to flush output to device */
};
```

- Added flush to allow for buffering
- Picolibc includes POSIX layer
  - requires read/write/lseek/open/close

printf & scanf

- float code takes a lot of space
  - can also drag in soft float & double code
- offer “int-only” and “float-only” versions
  - -DPICOLIBC_INTEGER_PRINTF_SCANF
  - -DPICOLIBC_FLOAT_PRINTF_SCANF

Using the float printf code

```
#define PICOLIBC_FLOAT_PRINTF_SCANF
#include <stdio.h>

int main(void)
{
    printf("%g\n", printf_float(355.0f/113.0f));
    return 0;
}
```

Comparing sizes (soft float)

```
$ size a*.out
   text     data      bss      dec      hex  filename
   2242       28        2     2272      8e0  a-int.out
   7920       28        2     7950     1f0e  a-float.out
  12904       28        2    12934     3286  a.out
```

Thread Local Storage

- TLS instead of 'struct reent'
- Linker limits TLS space to in-use vars
- RISC-V TLS support is excellent
  - Dedicated TLS base register
- Add API to set TLS base
  - To be used by an OS for thread switching
- Initial static TLS area setup by linker

crt0 and linker script

- Provide defaults for simple applications
  - User specifies RAM/ROM memories
- Allows configure tests to succeed
  - gcc hello-world.c
- Demonstrates requirements for more advanced users

semihosting

- Interface to host OS via debugger or QEMU
  - RISC-V version adapted from ARM version
- Console and file I/O
  - Printf debugging even before clocks are running
- _exit
  - Passes exit status through qemu
- RISC-V QEMU patches awaiting merge
  - QEMU just released 4.2.0

Testing

- newlib includes over 74000 tests
  - Thousands (and thousands!) fail
  - Not obviously used in decades
- picolibc has fixed these
  - All pass on RISC-V, ARM and x86 today
  - Testing 30 RISC-V combinations, along with ARM Cortex M3

hello-world.c

```c
#include <stdio.h>

int main(void)
{
  printf("hello, world\n");
  return 0;
}
```

Compiling

```
riscv64-unknown-elf-gcc
  -specs=picolibc.specs
  -march=rv32imac
  -mabi=ilp32
  -Thello-world.ld
  --oslib=semihost
  hello-world.c
```

Linker Script

```
__flash      = 0x80000000;
__flash_size = 0x00080000;
__ram        = 0x80080000;
__ram_size   = 0x00040000;
__stack_size = 1k;

INCLUDE picolibc.ld
```

Size

```
$ size a.out
   text    data     bss     dec     hex filename
    894      28       2     924     39c a.out
```

Running

```
qemu-system-riscv32
   -chardev stdio,id=stdio0
   -semihosting-config enable=on,chardev=stdio0
   -monitor none
   -serial none
   -machine spike,accel=tcg
   -cpu sifive-e31
   -kernel a.out
   -nographic
```

Demo

SiFive