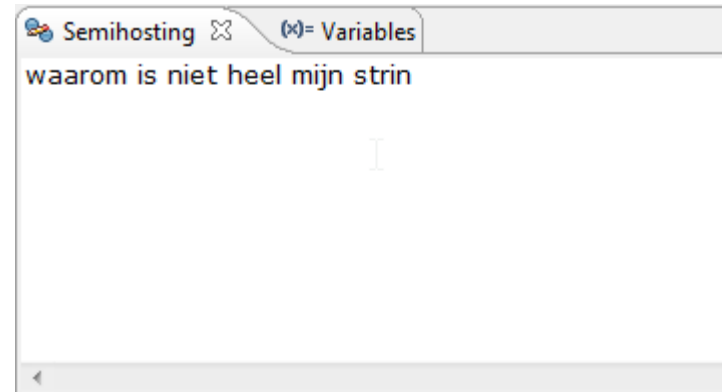


Printf en optimalisaties

Addendum op handson 3

Symptomen

```
1
2#include <stdio.h>
3
4int main(void)
5{
6    while(1)
7    {
8        printf("waarom is niet heel mijn string zichtbaar?\n");
9    }
10}
11
```



Output is gebufferd:

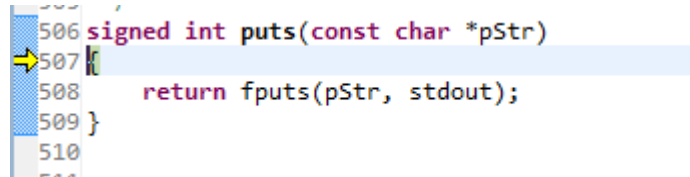
```
16 static char g_buf[16];
17 static char g_buf_len = 0;
18
19 /**
20  * @brief Transmit a char on semihosting mode.
21  *
22  * @param ch is the char that to send.
23  *
24  * @return Character to write.
25  */
26 void SH_SendChar(int ch) {
27     g_buf[g_buf_len++] = ch;
28     g_buf[g_buf_len] = '\0';
29     if (g_buf_len + 1 >= sizeof(g_buf) || ch == '\n' || ch == '\0') {
30         g_buf_len = 0;
31         /* Send the char */
32         if (SH_DoCommand(0x04, (int) g_buf, NULL) != 0) {
33             return;
34         }
35     }
36 }
```

- Buffer wordt pas geschreven
 - Buffer is vol
 - `ch == '\0'`
 - `ch == '\n'`

```
1
2 #include <stdio.h>
3
4 int main(void)
5 {
6     while(1)
7     {
8         printf("waarom is niet heel mijn string zichtbaar?\n");
9     }
10 }
11
```

Vaststelling 1:

- Bij het debuggen Breakpoint op printf en dan step into geeft:



```
506 signed int puts(const char *pStr)
507 {
508     return fputs(pStr, stdout);
509 }
510
```

— GEEN PRINTF MAAR PUTS!

Vaststelling 2:

- Wat indien er argumenten zijn met printf?

```
2 #include <stdio.h>
3
4 int main(void)
5 {
6     while(1)
7     {
8         printf("waarom is niet heel mijn string zichtbaar?\n",0);
9     }
10 }
11
```

Breakpoint,
step into:



```
463 signed int printf(const char *pFormat, ...)
464 {
465     va_list ap;
466     signed int result;
467
468     /* Forward call to vprintf */
469     va_start(ap, pFormat);
470     result = vprintf(pFormat, ap);
471     va_end(ap);
472
473     return result;
474 }
475
```

WEL PRINTF !!!

Semihosting Variables

waarom is niet heel mijn string zichtbaar?

Conclusie

- Iemand veranderd de printf naar puts en verwijderd hierbij de '\n'

NAME

puts - put a string on standard output

SYNOPSIS

```
#include <stdio.h>

int puts(const char *s);
```

DESCRIPTION

[CX] ⓘ The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of IEEE Std 1003.1-2001 defers to the ISO C standard. ⓘ

The *puts()* function shall write the string pointed to by *s*, followed by a <newline>, to the standard output stream *stdout*. The terminating null byte shall not be written.

[CX] ⓘ The *st_ctime* and *st_mtime* fields of the file shall be marked for update between the successful execution of *puts()* and the next successful completion of a call to *fflush()* or *fclose()* on the same stream or a call to *exit()* or *abort()*. ⓘ

!!!

Vaststelling 3:

- Printf met een string zonder `\n` achteraan en zonder argumenten wordt verzonden via `printf` en niet `puts!!!`

Conclusie

- Enige die dergelijke optimalisaties kan doen is de compiler. opzoekwerk geeft:
- http://www.cisellant.de/projects/gcc_printf/gcc_printf.html
- Oplossing 1:
 - Argument `-fno-builtin-printf` meegeven met de compiler

Configuration

Device	Compile	Link	Output	User	Debugger	Download
Compile						
Options						
FPU: <input type="text" value="Not use FPU"/>		Optimization: <input type="text" value="None (-O0)"/>				
Includepaths			Defined Symbols			
<input type="text" value="."/>			<input type="text" value="LM3S6965"/>			
<input type="button" value="Add"/>			<input type="button" value="Add"/>			

Nadeel oplossing 1

- Printf moet elk karakter in de string onderzoeken of dit een %-teken is.
- Echt probleem: Gcc doet zijn werk goed maar puts stuurt geen \n (volgens doc mag fputs geen \n sturen)

```
501 /**
502  * @brief Outputs a string on stdout.
503  *
504  * @param pStr String to output.
505  */
506 signed int puts(const char *pStr)
507 {
508     return fputs(pStr, stdout);
509 }
510
```

```
546 signed int fputs(const char *pStr, FILE *pStream)
547 {
548     signed int num = 0;
549
550     while (*pStr != 0) {
551         if (fputc(*pStr, pStream) == -1) {
552             return -1;
553         }
554         num++;
555         pStr++;
556     }
557
558     return num;
559 }
560
561
```

Oplossing 2

- Puts aanpassen zodat deze wel een \n stuurt:

```
506 signed int puts(const char *pStr)
507 {
508     return fputs(pStr, stdout);
509 }
```

```
506 signed int puts(const char *pStr)
507 {
508     signed int tmp;
509     tmp = fputs(pStr, stdout);
510     fputc('\n', stdout);
511     return tmp+1;
512 }
```