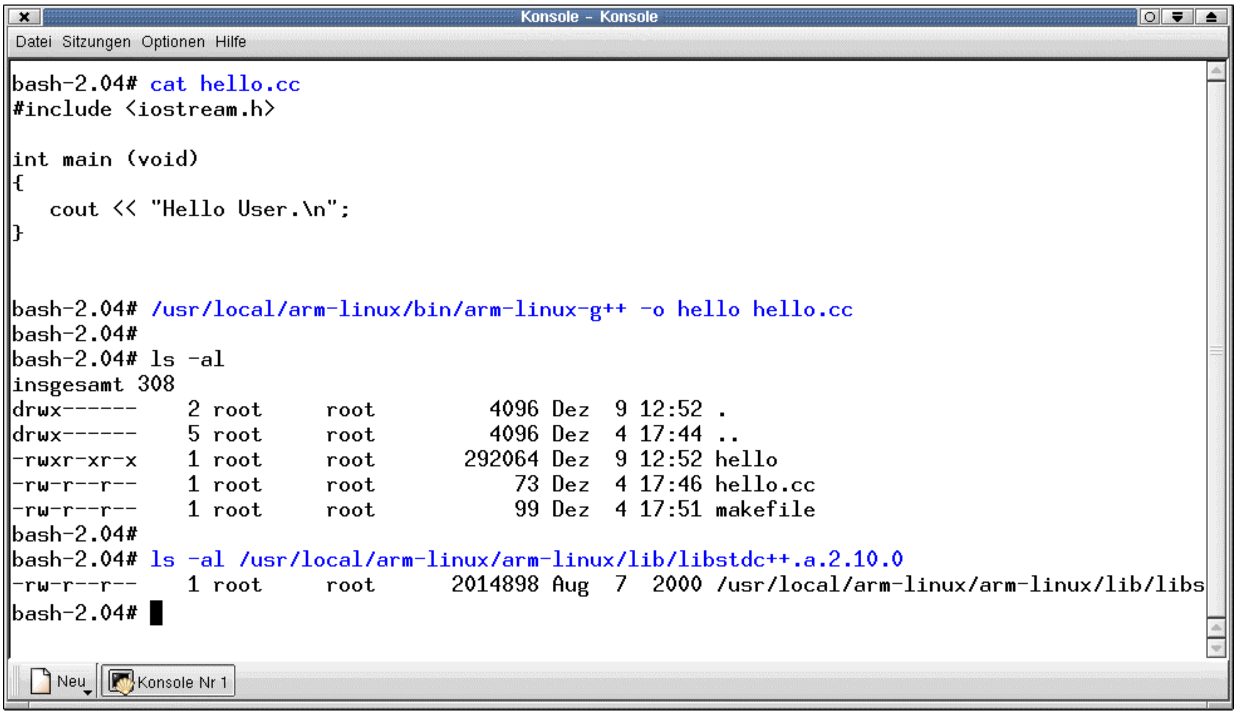
How to Program the DNP/1110 with C++

C++ programs need additional libraries. Please make sure that your DNP/1110 is using a Linux configuration with JFFS space. The document describes how to find out which libraries are necessary and how to install these libraries within the DNP/1110 JFFS file space.

- **1. Step**: Edit your C++ source code. Save your C++ source codes in files with the extension **.cc** (i.e. **hello.cc**).



```
bash-2.04# cat hello.cc
#include <iostream.h>

int main (void)
{
   cout << "Hello User.\n";
}


bash-2.04# /usr/local/arm-linux/bin/arm-linux-g++ -o hello hello.cc
bash-2.04#
bash-2.04# ls -al
insgesamt 308
drwx------    2 root     root         4096 Dez  9 12:52 .
drwx------    5 root     root         4096 Dez  4 17:44 ..
-rwxr-xr-x    1 root     root       292064 Dez  9 12:52 hello
-rw-r--r--    1 root     root           73 Dez  4 17:46 hello.cc
-rw-r--r--    1 root     root           99 Dez  4 17:51 makefile
bash-2.04#
bash-2.04# ls -al /usr/local/arm-linux/arm-linux/lib/libstdc++.a.2.10.0
-rw-r--r--    1 root     root      2014898 Aug  7  2000 /usr/local/arm-linux/arm-linux/lib/libs
bash-2.04# █
```

- **2. Step:** Run the Linux/GNU cross C++ compiler and build a executable for the DNP/1110 from your C++ source code file. The name of C++ cross compiler from the DNP/1110 starter Kit CD-ROM is **arm-linux-g++**. The following command line assumes that **hello.cc** is your C++ source code file and **hello** the name of the executable.

```
/usr/local/arm-linux/bin/arm-linux-g++ -o hello hello.cc
```

- **3. Step:** Transfer the new executable with FTP to the DNP/1110.

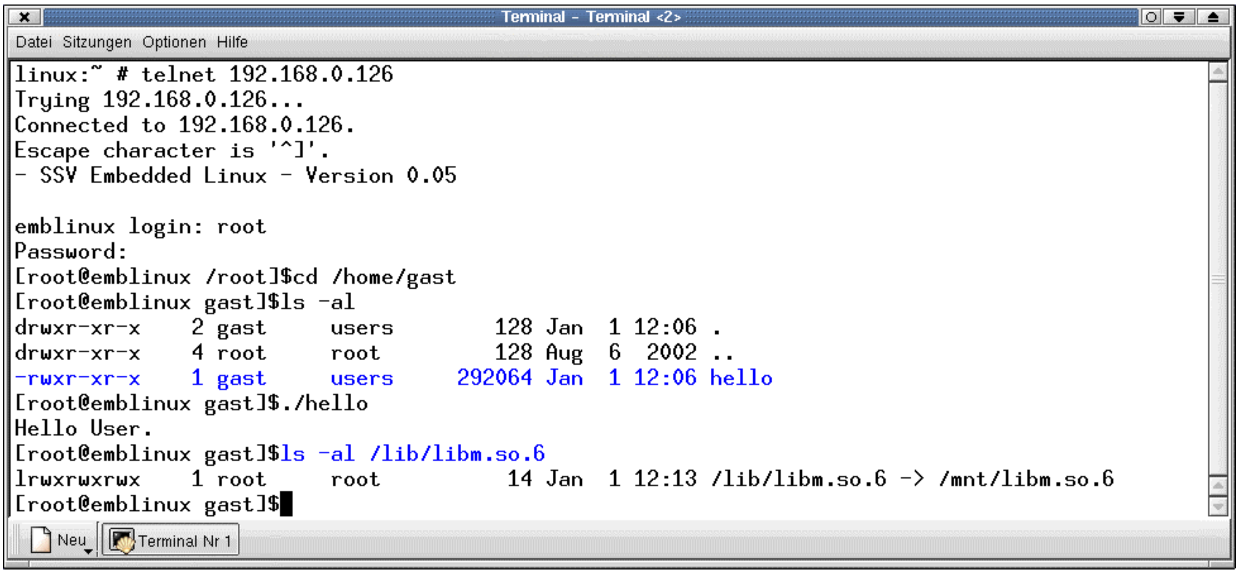
- **4. Step:** Try to run the **hello** executable on the DNP/1110. You get an error message for any missing library. In our case the executable **hello** is missing the library **libm.so.6**.

- **5. Step:** Transfer the missing library **libm.so.6** with FTP direct to the JFFS space (see DNP/1110 directory **/mnt**). Make sure that your FTP client owns the necessary write access rights. You find this library in the directory **/usr/local/arm-linux/arm-linux/lib** of your development system.

- **6. Step:** Build a symbolic link for the new libraries in **/mnt**. Setup a Telnet session to the DNP/1110. Make sure to get superuser rights for this session. Execute the following commands within your Telnet session.

  ```
  cd /lib
  ln -s /mnt/libm.so.6 libm.so.6
  ```

  The **ln** command builds in this case a link within the **/lib** directory of the DNP/1110 to the real library file in the **/mnt** directory (JFFS space of the DNP/1110).



```
linux:~ # telnet 192.168.0.126
Trying 192.168.0.126...
Connected to 192.168.0.126.
Escape character is '^]'.
- SSV Embedded Linux - Version 0.05

emblinux login: root
Password:
[root@emblinux /root]$cd /home/gast
[root@emblinux gast]$ls -al
drwxr-xr-x    2 gast     users         128 Jan  1 12:06 .
drwxr-xr-x    4 root     root          128 Aug  6  2002 ..
-rwxr-xr-x    1 gast     users      292064 Jan  1 12:06 hello
[root@emblinux gast]$./hello
Hello User.
[root@emblinux gast]$ls -al /lib/libm.so.6
lrwxrwxrwx    1 root     root           14 Jan  1 12:13 /lib/libm.so.6 -> /mnt/libm.so.6
[root@emblinux gast]$
```

- **7. Step:** Run the executable again. If your executable needs other or more libraries you get an error message. Repeat the steps for any additional library. Finally rebuild your RIMAGE.GZ to have the symbolic links permanent within **/lib**.

Please note: Within the directory **/usr/local/arm-linux/arm-linux/lib** of the development system you find also a static library with the name **libstdc++.a.2.10.0**. The g++ searches this library in step 2 for other C++ specific functions.