How to install the FLASH-Code V2.4 for Suse Linux 9.3

Andreas Maier

7. August 2008

1 Introduction

You must have `root`-permission to do the following steps:

2 Installing a Python Interpreter

If not already installed, just use YAST2 to install it.

3 Installing the Intel-Compilers Version 8 for C++ and Fortran90

- After registering on the Intel Web pages

  `http://www.intel.com/software/products/global/eval.htm`

  for the non-commercial unsupported version of the two compilers one receives two emails for each of the compilers including a Licence-File `*.lic`. Copy these two files to `/opt/intel/licenses` and check the permissions of the files and change them if necessary, so that other users of the group are allowed to read them:

  ```
  chmod g+r *.lic
  ```

- Set the enviroment variable

  ```
  export INTEL_LICENSE_FILE=/opt/intel/licenses
  ```

- Use `tar zxvf [filename]` to decompress and unpack the `*.tar.gz`-files to a temporary directory.

- Install the compilers by calling `./install.sh` in the temporary directory directory of each compiler. It is important to use `./`, otherwise one might call a different `install` existing in the `PATH`.[1]

  BUG: When changing the default installation path (`/opt/intel_cc_80`, `/opt/intel_fc_80`), there seems to be an bug in the installation routine. It is not possible to change the path to an directory like `/opt/intel/cc80`, that means an path more than two levels deep, for unkown reason.

- If other users than `root` should be able to use the compiler one has to change the read permission of the `*.lic`-files:

  ```
  chmod a+r [INSTALL_PATH]/licences/*.lic
  ```

- One has to add the compilers to the `PATH`. Therefore one can just copy the content of the scripts `[INSTALL_PATH_ICC]/bin/iccvars.sh` and `[INSTALL_PATH_IFORT]/bin/ifortvars.sh` to the file `.bashrc` in the home directory of each user or add the following lines to the file `\etc\bash.bashrc.local` (not `/etc/bash.bashrc`, because changes there will be lost during a system upgrade):

  ```
  . [INSTALL_PATH_IFORT]/bin/ifortvars.sh
  . [INSTALL_PATH_ICC]/bin/iccvars.sh
  . [INSTALL_PATH_IDB]/bin/idbvars.sh
  ```

  This ensures, that every user on the local computer has the compilers in his `PATH`.

  BUG: To prevent the output `"What manual page do you want?"` when logging into the localhost and executing a command, e.g.: `ssh localhost date`, edit the files `ifortvars.sh` and `iccvars.sh` by replacing `$(man -w)` with `$(manpath)`. This is necessary, because otherwise you will get this message for every process started by `mpirun`.

4 Installing MPICH 1.2.5.2 for the use with the Intel compilers

4.1 Configuring SSH2

One has to configure SSH2 in such a way, that one can login to the `localhost` without typing a password. Otherwise one would have to provide an password for each process MPI is starting. Each user, who wants to use MPI has to do the following in his home directory:

- Create your authentication key with `ssh-keygen -t rsa`.
- Copy the generated file `".ssh/id_rsa.pub"` to `.ssh/authorized_keys2`.

That should do it. For more information have a look at the MPICH documentation `mpichman_chp4.pdf`.

[1] The installer might confuse you at one point, there ist seems you can only choose options `1a-1e`. However, you really have to type `1` for the installation to proceed.

4.2 Configuring and making of MPICH 1.2.5.2

- Use `tar zxvf mpich.tar.gz` to decompress and unpack the file to a temporary directory.
- Copy the content of this directory to `/usr/local/mpich-1.2.5.2` if you want to install it there.
- Make sure `ifort` and `icc` is in your `PATH`.
- Start configuring with the following commandline (assuming you are in `/usr/local/mpich-1.2.5.2`):

  ```
  ./configure --prefix /usr/local/mpich-1.2.5.2 -cc=icc
  -c++=icc -fc=ifort f90=ifort -rsh=ssh | tee config.log
  ```

- Make MPICH with `make`.
- Every user who wants to use MPI has to add `mpirun` to the `PATH` by adding the following line to `.bashrc`:

  ```
  export PATH=$PATH:/usr/local/mpich-1.2.5.2/bin
  ```

  (Alternatively one can add this line to `/etc/bash.bashrc.local`, so all users on the local computer have MPI in their `PATH`)

- Build and run two simple test programs:

  ```
  cd examples/basic
  make cpi
  make fpi
  ../../bin/mpirun -np 4 cpi
  ../../bin/mpirun -np 4 fpi
  ```

  GOOD LUCK!

5 Installing of HDF5

5.1 Obtaining the necessary packages

Because the developers of HDF5 do not support Linux Kernel $>$ 2.4, the precompiled binaries of HDF5 that are available from

`http://hdf.ncsa.uiuc.edu/HDF5/release/obtain5.htm`

do not work with Suse 9.3 (Linux Kernel 2.6, gcc 3.3.5). So one has to compile HDF5 from the sources on its own. But HDF5 V1.6.1 does not compile under Suse 9.3; `gcc` 3.3.5 generates several serious bugs. In HDF5 V1.6.4 and higher the hyperslab selection APIs in HDF5 (`H5Sselect_hyperslab`, `H5Sselect_elements`) have changed so that the offset parameter is of type "`hsize_t`"rather than "`hssize_t`änd therefore some programs are incompatible with the newest versions of HDF5. That why one better obtains version 1.6.3 from HDF5. It is available from

`ftp://ftp.ncsa.uiuc.edu/HDF/HDF5/prev-releases/hdf5-1.6.3.tar.gz`

In addition one needs version 1.2 of SZIP for compiling HDF5 1.6.3. It is sufficent to get the precompiled version for Linux Kernels 2.4 from

`ftp://ftp.ncsa.uiuc.edu/HDF/szip/1.2/bin/linux/szip-linux2.4.tar.gz`

The new version 2.0 of SZIP is not supported in HDF5 1.6.3. In addition version 2.0 of SZIP is a shared library by default (in 1.2 it was static library) which causes aditional trouble with paths when trying to compile and run programs. For these reasons version 2.0 of SZIP should be avoided at the moment.

5.2 Compiling HDF5

Before compiling HDF5 one should install SZIP 1.2 at some place. For example:

```
> cp szip-linux2.4.tar.gz /opt2
> cd /opt2
> tar zxvf szip-linux2.4.tar.gz
```

Then decompress the source code of HDF5 at some place

```
> tar zxvf hdf5-1.6.3.tar.gz
```

change into the decompressed directory and type the following commands to compile HDF5 with support for SZIP:

```
> ./configure --prefix=/opt2/hdf5-1.6.3 --with-szlib=/opt2/szip-linux2.4
> make check >& check.log
> make install
```

The option `--prefix` gives the path where the compiled binaries and libraries should be installed to. The option `--with-szlib` tells HDF5 to compile with SZIP support and looks for the SZ-library at the given path.

Because HDF5 is compiled as a shared library you have to make sure that executables find the library during runtime and you should add the following line to your `.bashrc`

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt2/hdf5-1.6.3/lib
```

This makes sure that the HDF5 libraries are in your library search path.

6 Installing of IDL 6.0

BUG: xflash (see below) seems to be not working with IDL 6.0. It was tested succesfully only with IDL 5.6.

IDL is a commercial package. A free trial version that runs for seven minutes is available at `www.rsinc.com` or directly from
`ftp://download.rsinc.com/pub/idl_6.0/unix/www/linux.x86`

- Download the file `idl_linux.x86.tar.gz` and use `tar zxvf idl_linux.x86.tar.gz` to decompress and unpack the file to the directories `idl6.0`, `licenses`.
- Generate a new directory `/usr/local/rsi` and give access to it for every user by `chmod a+rx /usr/local/rsi`.
- Copy the directories `idl6.0` and `licenses` and the script `install.sh` to `/usr/local/rsi`.
- Start `./install.sh` from `/usr/local/rsi` and follow the instructions (Therefore also have a look at the download and installation instructions of IDL in `/idl6.0/help/instlic.pdf`).

7 Installing, making and testing of the Flash Code Version 2.4

You don't need `root`-permission to install the Flash Code. The code is available on request from `http://flash.uchicago.edu`. One can download the code only once using a single specific IP. Download the file `FLASH2.4.tar.gz`.

- Use `tar zxvf FLASH2.4.tar.gz` to decompress and unpack the file to a temporary directory. Copy the content of this directory to a place of your choice, e.g. `[HOME-DIRECTORY]/FLASH2.4`.
- Add two enviroment variables to your `.bashrc`:

  ```
  export XFLASH_DIR="[HOME-DIRECTORY]/FLASH2.4/tools/fidlr2"
  export IDL_PATH="$XFLASH_DIR:$IDL_PATH"
  ```

- You also need to setup the wrappers that communicate between IDL and the HDF5 library. Therefore go to `[HOME-DIRECTORY]/FLASH2.4/tools/fidlr2` and edit the file `Makefile.linux`. Change the paths of `HDF5path` and `IDLpath` to the directories where IDL and HDF5 are installed on your machine, e.g.:

  ```
  HDF5path = /opt2/hdf5-1.6.3
  IDLpath = /usr/local/rsi/idl/external
  ```

  Add a new line defining the path to the szlib:

```
SZLIBpath= /opt2/szip-linux2.4
```

Change the `LIB` line to:

```
LIB = -L${HDF5path}/lib -L${SZLIBpath}/lib -lhdf5 -lz -lsz
```

To generate the shared-object library `h5_wrappers.so` start:

```
gmake -f Makefile.linux
```

- Change to the directory `/FLASH2.4/source/sites/Prototypes/Linux/`. Rename the file `Makefile.h` to `Makefile.h.port` (for Portland-Compiler) and rename `Makefile.h.ifc` to `Makefile.h`. Then edit this file and change the path to the HDF5 directory, e.g.:

  ```
  HDF5_PATH = /opt2/hdf5-1.6.3/
  ```

  Add a new line defining the path to the szlib:

  ```
  SZLIB_PATH= /opt2/szip-linux2.4
  ```

  Also change the line of `LIB_HDF5` to :

  ```
  LIB_HDF5 = -L$(HDF5_PATH)/lib -L$(SZLIB_PATH)/lib -lhdf5 -lz -lsz
  ```

  Change the `FFLAGS_OPT`[2] to

  ```
  FFLAGS_OPT = -c -r8 -i4 -O3 -static
  ```

Now we are ready to start a first test. Following the documentation of the Flash code:

- From the directory `/FLASH2.4` configure the FLASH source tree for the Sedov explosion problem:

  ```
  ./setup  sedov -auto
  ```

- Execute `gmake`.
- Generate a new parameter file `flash.par` as described in the documentation (`FLASH2.4/docs/flash/flash2_ug.pdf.gz`). Do not use the file `FLASH2.4/object/flash.par` generated by `./setup`.
- Assuming compilation and linking were succesful, you should find an executable named `flash2` in the `/object` directory. To run FLASH on 2 processors, type

[2] Do not use `-fast` with ifort. `-fast` means `-O3 -static -ipo` and FLASH doesn't compile with `-ipo`.

```
mpirun -np 2 object/flash2 flash.par
```

- to make sure the plots with `xflash` have a white background (even on a true color display) generate or change the file `.idlstartup` in our home directory like

```
; Commands to be executed every time IDL is started
;
print,'Hi!'
!EDIT_INPUT = 200
; to circumvent problems on 24-bit linux system
;.run ~/idl/pro/colorset
;colorset,retain=2,decomposed=0,/pseudocolor
; this seems to work as well
if(!version.os_family eq 'unix') then device, true_color=24
window, /free, /pixmap, colors=-10
wdelete, !d.window
device, retain=2, decomposed=0, set_character_size=[10,12]
device, get_visual_depth=depth
print, 'Display depth: ',depth
print, 'Colour table size: ',!d.table_size
; alternatively, you can run in 24-but mode, but need to properly
; address colours as such
; ****************** alternatively
red    = [255,0,0]
green  = [0,255,0]
blue   = [0,0,255]
;
yellow = [255,255,0]
cyan   = [0,255,255]
purple = [255,0,255]
;
```

  This is necessary, because IDL 5.6 behaves strange when confronted with a 24-bit linux system[3]

- Run IDL (`idl`).

- To make sure the plots will be correctly refreshed change the Parameter `RETAIN` of the `DEVICE`. Therefore type

```
IDL> DEVICE, RETAIN='2'
```

  at the commandline. Another solution is to use `idlde` instead of `idl` and change the graphics preferences of *backing store* to *Pixmap* instead of *System*.

[3] Thanks to Marcus Brüggen for pointing this out.

- Enter `xflash` at the `IDL>` prompt.
- Select the output file `~/FLASH2.4/sedov_4_hdf5_plt_cnt_0000` through the *File/Open Prototype...* dialog box.

...to be continued