查看完整版本: [-- linux下常见的Fortran编译器介绍 --]

popleaf1 2008-06-21 21:52

在各种**Linux**平台下，常用的有下列几种：**g77(f77)，Intel Fortran compiler, G95, gfortran**.现在对之一一做介绍：
1、g77(f77)：是GCC中默认的**fortran编译器**，编译出的程序执行速度快，健壮，是十分优秀的**编译器**，可惜只能针对f77格式的**fortran**代码；
下面就其安装做一简单介绍：
[日期：2008-03-23] 来源：**Linux**公社 作者：**Linux**整理

在**Linux**下安装g77 **fortran** complier的具体过程：

1.至ftp://ftp.ntu.edu.tw/pub/gnu/gnu/g77下载g77-0.5.23.tar.gz
至ftp://ftp.ntu.edu.tw/pub/gnu/gnu/gcc下载gcc-2.8.1.tar.gz
确定这两个东西是相容的(g77-0.5.23.tar.gz跟gcc-2.8.*.tar.gz等版本相容)
可以先解压g77-0.5.*.tar.gz然后查看解压后资料夹内的./f/INSTALL档案查看跟它相容的gcc版本.

2.用root的身分在/usr/下制造一个叫FSF的目录,如以下指令
#cd /usr
#mkdir FSF
将下载好的两个压缩档移到FSF目录中
#mv g77-0.5.23.tar.gz /usr/FSF
#mv gcc-2.8.1.tar.gz /usr/FSF

3.接下来跟着以下指令一步一步做,不要改变任何细节:
#cd /usr/src
#gunzip -c < /usr/FSF/gcc-2.8.1.tar.gz | tar xf - (注意|是pipe)
#gunzip -c < /usr/FSF/g77-0.5.23.tar.gz | tar xf -
#ln -s gcc-2.8.1 gcc
#ln -s g77-0.5.23 g77
#mv -i g77/* gcc
#cd gcc
#./configure –prefix=/usr
#make bootstrap (这里请耐心等它跑完这边最容易出错)
#make compare
#rm -fr stage1
#make -k install
#g77 -v (检查g77版本确定已安装OK)

4.详细说明请参看g77-0.5.23.tar.gz解压后的./g77-0.5.23/f/INSTALL档

popleaf1 2008-06-21 21:57

2、Intel **Fortran** Compiler，这个**编译器**功能十分强大，兼容性也不错，对f77、**f90**格式的代码均可以编译，同时性能也很好，只是对某些f77格式的代码在编译时有些问题，在**f77编译器**下可以，但在ifort下就不行了，同时在编译**c-fortran** 接口程序时也会因为代码的不兼容，出现各种问题，下面就其安装做一简单介绍：
Intel **FORTRAN 编译器** 入门系列之一: **Linux** 安装和使用
csdn, author: intel_iclifort

Intel **FORTRAN 编译器**能支持安装在绝大多数的主流**Linux**发行版本, 包括 Asianux* 3.0, Debian* 4.0, Red Hat

Enterprise **Linux*** 3, 4, 5, Fedora* 7, SUSE **LINUX** Enterprise Server* 9, 10, TurboLinux* 11, Ubuntu 7.0等等

I. Intel **FORTRAN 编译器**安装

1) 下载安装包后, 解包, 并运行安装脚本 (请尽量使用 root 权限的账号进行安装)
> tar -zxvf l_fc_x_10.1.xxx.tar.gz
> cd l_fc_x_10.1.xxx
> ./install.sh

2) 选择 1 进行安装, 并提供许可文件(License File). 注意请输入完整的全路径, 包括许可文件名 (许可文件通常以.lic结尾, 建议放入缺省目录/opt/intel/licenses)

3) 选择 1 进行典型安装 (Typical Install)

4) 根据提示, 阅读许可, 选择安装路径等等, 直到全部结束

II. Intel **FORTRAN 编译器**使用

注意, 缺省的安装目录在 /opt/intel/fc[e]/xx.x.xxx/ (xx.x.xxx代表版本号, fc代表IA-32 and IA-64版本, fce代表Intel 64版本)

使用前, 需要设置相关的环境:
] source /opt/intel/fc/10.1.xxx/bin/ifortvars.sh (或者是ifortvars.csh)

然后编译源文件:
] ifort my_source_file.**f90**

查看当前版本
] ifort -V

参看支持的所有命令行选项
] ifort -help

III. 常见问题

Q: 如果碰到安装失败, 如何解决 ?
A: 首先, 请确认你下载了最新的发行版本, 并检查当前系统,
1) 系统是否已经安装 **Linux** Developer tools 选件, 包括 GCC, G++ 和其它相关的开发工具包
2) 系统是否已经安装 **Linux** 选件 compat-libstdc++, 它提供 libstdc++.so.5 库
3) 如果是Intel 64(EM64T)环境, 系统是否已经安装了 32-bit 库 (可能被称作 ia32-libs)

然后, 查看发行说明(Release Notes), 核对你的系统是否支持
最后, 联系Intel Premier Support (http://premier.intel.com), 寻求帮助
当然, 还可以通过论坛, 搜索网络, 和他人讨论

Q: 使用时, 遇到错误信息 "ifort: error: could not find directory in which g++ resides"
A: Intel **Fortran编译器**无法在你的系统中找到GNU* g++ **编译器**. 可能是由于你没有安装 GCC 开发包, 或者 g++ 不是安装在缺省路径, 或者你使用了非英文的**Linux**版本. 解决办法请访问Intel网站:
http://support.intel.com/support/performancetools/**fortran**/**linux**/sb/CS-017386.htm

Q: 使用时, 遇到错误信息 "Intel 10.x compiler's dependency on /usr/lib/libstdc++.so.5"
A: Intel 10.x **编译器**为了保证和基于 GCC 3.2 的系统兼容, 需要使用标准 C++ 库 /usr/lib/libstdc++.so.5, 但是很多比较新的 **Linux** 发行版本中开始使用 GCC 3.4, 并且提供了全新的标准 C++ 库 /usr/lib/libstdc++.so.6. 因此需要安装 compat-libstdc++ RPM包, 它包含了 /usr/lib/libstdc++.so.5 库.

IV. 常用链接:

Intel **Linux FORTRAN 编译器** 帮助文档: http://www.intel.com/cd/software/products/asmo-na/eng/346152.htm

Intel **Linux FORTRAN 编译器** 发行说明:

http://www.intel.com/software/products/compilers/docs/flin/release_notes.htm
Intel **Linux FORTRAN 编译器** 安装指导:
http://www.intel.com/software/products/compilers/docs/flin/install.htm
Intel **Linux FORTRAN 编译器** 英文 FAQ: http://www.intel.com/cd/software/products/asmo-na/eng/346192.htm

popleaf1 2008-06-21 22:02

G95:这个**编译器**我用的不多，仅仅用过几次，这里做一简单介绍：
其使用十分方便，到http://www.g95.org/下载一个可执行包，如果愿意，也可以下载代码包进行编译安装，我使用时是在CentOS4.5下直接在bin目录下建立了主程序的快捷方式，就直接可以用了，具体大家请看代码包里面Readme帮助文件，在我接触的**Fortran编译器**里面这是最简单的，对代码的兼容性不错，生成的程序的健壮性也还可以，但在使用中发现，对C-**fortran**接口的链接生成可执行程序时会发生内存的偶尔泄漏，与G95组织联系，给出的答案也很模糊，由于时间关系，我没有深入研究，如果哪位同仁有时间，请帮忙补充，在这里popleaf1先谢谢。

popleaf1 2008-06-21 22:04

最近十分的繁忙，同事们都有事，我只能代劳，所以关注论坛的时间相对少了一些，敬请大家见谅。

MCyongshi 2008-06-22 11:30

popleaf1斑竹辛苦了,以后还要向您多多学习,现在我的编程水平太差了,呵呵 [s:5]

popleaf1 2008-06-22 18:34

在 GCC 4.0 之前，g77 是 GCC 的一部分；此后，gfortran 是 GCC 的一部分。g95 是一个基于 GCC 的 **Fortran 编译器**，它不是 GCC 的一部分。

g77介绍
g77 是 **Fortran**77 的**编译器**。它对 **Fortran** 77 标准提供完备的支持，并支持 **Fortran** 90 和 95 的部分特性。
由于 **Fortran** 77 标准在数值计算中的影响力，g77 可能是应用最广的**Fortran编译器**。
在 GCC 4.0 之前，g77 是 GCC 的一部分，但现在，g77 已经停止开发。
g77为何不再被支持：

gcc-4.0 改变了 gcc 中所有语言的前端界面。由于缺少志愿者和公司来更新 g77 到 gcc-4.0 的架构，因此它被废弃了。不同于 g77，gfortran 项目处于活跃开发期，因此它 取代了 g77 的位置。

这是一篇g77使用入门(**http://wiki.ubuntu.org.cn/index.php?title=Compiling_Fortran77&variant=zh-cn**)

gfortran介绍
GNU 的 **Fortran** 95 **编译器**，支持**Fortran**95和一部分**Fortran**2003的功能。
取代 g77 集成在 GCC 4.0 及以后版本中
这是一篇gfortran使用入门(**http://wiki.ubuntu.org.cn/index.php?title=Compiling_Fortran&variant=zh-cn**)

关于g95
gfortran 不是 g95：

gfortran 是一个 **Fortran** 95 的**编译器**，它是 GCC 的一部分。
g95 是另一个 **Fortran** 95 的**编译器**，它是一个基于 GCC 的**编译器**。
历史：

Andrew Vaught 在 2000 年上半年创建了 g95——一个使用 GCC 做后端的开放源代码的 **Fortran** 95 **编译器**。在随后的两年里，这是一个多人协作的项目，但是 2002 年下半年 Andrew Vaught 决定单独开发 g95。2003 年 1 月，gfortran 项目创建，它建立在当时 GPL 授权的 g95 源码的基础上，目的是允许协同开发并与 GCC 代码集成。

从那时起，Andrew 一个人在持续地开发 g95，g95 与 gfortran 的差别也越来越大。因此，gfortran 项目组也无法为 g95 提供支持或建议。

popleaf1 2008-06-25 14:03

Fortran 编程中相关文件后缀
.a 静态库 (archive)
.f, .for, .FOR

.ftn*, .**f90***, .f95*, .f03* **Fortran**源代码（不需编译预处理）
.F, .fpp, .FPP
.FTN*, .**F90***, .F95*, .F03* **Fortran**源代码（需要编译预处理）
.r **Fortran**源代码（需要RatFor编译预处理）
.o 对象文件
.s 汇编语言代码
.so 动态库

其中，标 * 的后缀名是gfortran的文件后缀，g77不能识别。

单个源文件生成可执行程序
传统的 **Fortran** 程序（也就是以 **Fortran** 77 为代表的）只能用大写字符书写，而且每行前六个字符为特定用途所保留。第一列为字符 C 或 * 所保留，用来表征整行都是注释。第二列到第六列是为标号预留的。代码从第七列开始，到72列结束（73列及以后将被直接忽略，可作注释）。下面是示例程序采用的是传统的 **Fortran** 格式：

```
C  helloworld.f
C
      PROGRAM HELLOWORLD
      WRITE(*,10)
  10 FORMAT('hello, world')
      END PROGRAM HELLOWORLD
```

编译器 gortran 并不要求所有代码都大写——程序中任何关键词都可以用小写字母。下面的命令将该程序编译成可执行文件：

```
$ gfortran helloworld.f -o helloworld
```

注意到：gfortran 默认会将 .f, .for, .fpp, .ftn, .F, .FOR, .FPP 和 .FTN 结尾的文件作为固定格式处理，而将.**f90**, .f95, .f03, .**F90**, .F95 和 .F03 结尾的文件作为自由格式来处理。如果我们将上面程序文件重命名为 helloworld.**f90**，那么我们必须手动指定其为固定格式：

```
$ mv helloworld.f helloworld.f90
$ gfortran helloworld.f90 -ffixed-form -o helloworld
```

Fortran 90及以后的标准允许并鼓励用自由的格式书写 **Fortran** 代码。注释以感叹号（！）开始直到行尾。先前的程序采用自由格式重写如下，其中语句、标号都可从任一列开始：

```
! helloworldff.f90
!
Program Helloworld
write(*,10)
10 format('hello, world')
       end Program Helloworld
```

后缀名为 .**f90**，故 gfortran 将其作为自由格式处理

```
$ gfortran helloworldff.f90 -o helloworldff
```

同样，如果将程序重命名为传统后缀名，那么要通过在命令行中加入选项 -ffree-form 进行编译，如下：

```
$ mv helloworldff.f90 helloworldff.for
$ gfortran -ffree-form helloworldff.for -o helloworldff
```

由于两种格式的具有很大的区别，程序书写是只能选择其中的一种格式进行书写。注意：遵守后缀约定是很重要的。

多个源文件生成可执行程序
命令 gfortran 可将多个 **fortran** 源码文件编译链接成为一个单一的可执行程序。下面列出了一个保存在文件 caller.f 中的简单程序的主体部分，它调用一个函数并显示出结果：

```
C  caller.f
C
      PROGRAM CALLER
      I = Iaverageof(10,20,83)
      WRITE(*,10) 'Average=', I
  10 FORMAT(A,I5)
      END PROGRAM CALLER
```

名为 Iaverage 函数定义在另一个独立的源文件中，如下：

```
C  called.f
C
      INTEGER FUNCTION Iaverageof(i,j,k)
      Iaverageof = (i + j + k) / 3
      RETURN
      END FUNCTION Iaverageof
```

通过下面的语句这两个源码文件可被编译链接成一个名为 caller 的可执行程序：

```
$ gfortran caller.f called.f -o caller
```
同样的结果可由下面的命令序列得到——先将每一个源码文件编译成对象文件，而后将对象文件链接为可执行程序：

```
$ gfortran -c caller.f -o caller.o
$ gfortran -c called.f -o called.o
$ gfortran caller.o called.o -o caller
```
生成汇编代码
选项 -S 指示**编译器** gfortran 生成汇编语言代码然后结束。要得到我们本文先前的 helloworld.f 例子的汇编代码，只需输入以下命令：

```
$ gfortran -S helloworld.f
```
生成的汇编语言文件名为 helloworld.s。汇编语言的具体形式依赖于**编译器**的目标平台。

编译预处理
编译以 .F, .fpp, .FPP, .FTN, .**F90**, .F95 和 .F03 结尾的文件时，在它真正编译之前需要预处理。预处理器原本是为协助 C 语言工作所设计的。下面的例子是一个自由格式的 **Fortran** 程序，它通过预处理器将一个函数包含进主程序：

```
! evenup.F90
!
#define ROUNDUP
#include "iruefunc.h"
!
program evenup
do 300 i=11,22
   j = irue(i)
   write(*,10) i,j
300 continue
10 format(I5,I5)
end program evenup
```
函数 irue() 的源代码保存在文件 iruefunc.h 中，根据宏 ROUNDUP 所定义的值的不同将产生不同的编译结果。该函数将任何一个奇数近似为一个偶数。默认情况下，它向下取近似，但是当 ROUNDUP 被定义时，该函数将向上取近似而得到一个偶数。ireu() 的函数体如下：

```
integer function irue(i)
k = i / 2
k = k * 2
if (i .EQ. k) then
   irue = i
else
#ifdef ROUNDUP
   irue = i + 1
#else
   irue = i - 1
#endif
end if
end function irue
```
下面的命令将该程序编译成可执行文件：

```
$ gfortran evenup.F90 -o evenup
```
采用自由格式写程序以利用预处理器不是必须的。固定格式程序也可进行编译预处理，下面的程序也是有效的：

```
C adder.F
C
#define SEVEN 7
#define NINE 9
C
     program adder
     isum = SEVEN + NINE
     write(*,10) isum
  10 format(I5)
     end program adder
```
下面的命令将该程序编译成可执行文件：

```
$ gfortran adder.F -o adder
```
理解gfortran是gcc的前端
像 g++ 一样，gfortran 也只是设置过 **Fortran** 程序所需基本环境的 gcc 的一个前端。本文一开始的例子我们可以通过下面 gcc 的命令来编译：

$ gcc helloworld.f -o helloworld -lgfortran -lgfortranbegin
库文件 libgfortranbegin.a (通过命令行选项 -lgfortranbegin 被调用) 包含运行和终止一个 **Fortran** 程序所必须的开始和退出代码。库文件 libgfortran.a 包含 **Fortran** 底层的输入输出等所需要的运行函数。当运行 gfortran 时，会自动链接这两个库。这和下面的命令是等价的：

$ gfortran helloworld.f -o helloworld
当我们运行 gfortran 时，实际上运行并不是这个**编译器**，而是**编译器**驱动器。该驱动器解析命令行中所给出的选项，然后才调用真正的**编译器**，汇编器和链接器。默认情况下，**编译器**驱动器根据命令行中给定的文件的后缀决定它自己下一步的动作：一个名为 foo.c 将传递给 C **编译器**，而名为 foo.f95 的文件将传递给 **Fortran** 95 的**编译器**，等等。

理解了这一点，我们就可以知道 gcc helloworld.f 将自动调用 **fortran** 的**编译器**。只不过我们要为链接器指定必要的库。

理解了这一点，我们可以知道 gfortran helloworld.c 可以编译一个 c 程序，gfortran helloworld.cpp -lstdc++ 编译的是一个 C++ 程序。

查看完整版本: [-- linux下常见的Fortran编译器介绍 --] [-- top --]