编译器和运行时库

李会民
hmli@ustc.edu.cn

中国科学技术大学 超级计算中心

2014-07-18

1 Intel C/C++、Fortran编译器简介
- Intel编译器安装配置
- Intel C/C++编译器用法
- Intel Fortran编译器用法
- Intel调试器用法

2 GNU C/C++、Fortran编译器简介
- GNU C/C++编译器用法
- GNU Fortran编译器用法
- GNU程序调试器用法

3 数学函数库
- Intel MKL数学函数库
- 其余数学统计函数库

4 联系信息

Intel C/C++、Fortran编译器简介

Intel C/C++ Fortran编译器是一种主要针对Inetl CPU平台的高性能编译器，在AMD Opteron平台上性能也不错，可用于开发复杂且要进行大量计算的程序。

Linux版本的Intel C/C++ Fortran编译器在X86_64（又称Intel 64、EM64T或AMD64）平台上默认安装路径：

- 13、14系列：
  - */opt/intel/composer_xe_2013.2.146*：真正目录
  - */opt/intel/composer_xe_2013*：上述目录的链接
  - */opt/intel/composerxe*：上述目录的链接
- 12系列：
  - */opt/intel/composerxe-2011.3.174*
  - */opt/intel/composer_xe_2011_sp1.6.233*
- 11系列：*/opt/intel/Compiler/y/z*（如安装的为11.0.081，则对应的y为11.0，z为081）
- 10系列：C/C++编译器默认安装在*/opt/intel/cce/x*；Fortran编译器在*/opt/intel/fce/x*（x为版本号，如10.1.018）
- 用户可以查看*/opt/intel*类似目录下还安装有哪些版本

1 Intel C/C++、Fortran编译器简介
- Intel编译器安装配置
- Intel C/C++编译器用法
- Intel Fortran编译器用法
- Intel调试器用法

2 GNU C/C++、Fortran编译器简介
- GNU C/C++编译器用法
- GNU Fortran编译器用法
- GNU程序调试器用法

3 数学函数库
- Intel MKL数学函数库
- 其余数学统计函数库

4 联系信息

Intel编译器下载

- 编译器下载主页：https://software.intel.com/en-us/intel-sdp-home
- 点击 Complete Products List
- 选择所需要的产品，比如：Intel® Cluster Studio for Windows and Linux[1]
- 点击右上角的Or Download a Free 30-Day Evaluation Version
- Download Linux version
- 按照提示注册，注册成功后登录注册邮箱查看注册信件，按照提示下载
- 在打开的页面中，注意保存序列号，安装产品时将需要使用它

[1]此版本含C/C++ Fortran编译器、MKL数值函数库及MPI等

Intel编译器安装 I

- 解压缩: *tar␣xvf␣l_ics_2013.1.046.tgz*
- 安装: *./install.sh*
- 步骤1: 回车

```
Step 1 of 7 | Welcome
--------------------------------------------------------------------------------
Welcome to the Intel(R) Cluster Tools 2013 SP1 Update 1 for Linux* OS
installation program.


--------------------------------------------------------------------------------
You will complete the steps below during this installation:
Step 1 : Welcome
Step 2 : License agreement
Step 3 : Activation
Step 4 : Intel(R) Software Improvement Program
Step 5 : Options
Step 6 : Installation
Step 7 : Complete

--------------------------------------------------------------------------------
Press "Enter" key to continue or "q" to quit:
```

Intel编译器安装 II

- 步骤2：某些依赖关系未满足，酌情处理，或回车

```
Step 1 of 7 | Welcome > Missing Optional Prerequisite(s)
--------------------------------------------------------------------------------
There are one or more optional unresolved issues. It is highly recommended to
resolve them all before you continue the installation. You can fix them without
exiting from the installation and re-check. Or you can quit from the
installation, fix them and run the installation again.
--------------------------------------------------------------------------------
Missing optional prerequisites
-- Intel(R) Fortran Composer XE 2013 SP1 Update 2 for Linux*: Unsupported OS
-- Intel(R) C++ Composer XE 2013 SP1 Update 2 for Linux*: Unsupported OS
--------------------------------------------------------------------------------
1. Skip missing optional prerequisites [default]
2. Show the detailed info about issue(s)
3. Re-check the prerequisites

h. Help
b. Back to the previous menu
q. Quit
--------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]:
```

Intel编译器安装 III

- 同意协议：accept回车

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

Do you agree to be bound by the terms and conditions of this license agreement?
Type 'accept' to continue or 'decline' to go back to the previous menu:

Intel编译器安装 IV

- 激活：选择合适方式，比如选择2，输入序列号，回车

```
Step 3 of 7 | Activation
--------------------------------------------------------------------------------
If you have purchased this product and have the serial number and a connection
to the internet you can choose to activate the product at this time. Activation
is a secure and anonymous one-time process that verifies your software licensing
rights to use the product. Alternatively, you can choose to evaluate the product
or defer activation by choosing the evaluate option. Evaluation software will
time out in about one month. Also you can use license file, license manager, or
remote activation if the system you are installing on does not have internet
access activation options.
--------------------------------------------------------------------------------
1. Use existing license [default]
2. I want to activate my product using a serial number
3. I want to evaluate my product or activate later
4. I want to activate either remotely, or by using a license file, or by using a
license manager

h. Help
b. Back to the previous menu
q. Quit
--------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]: 2
```

Intel编译器安装 V

- 序列号方式激活：输入serial number回车

```
Step 3 of 7 | Activation
--------------------------------------------------------------------------------
If you have purchased this product and have the serial number and a connection
time out in about one month. Also you can use license file, license manager, or
remote activation if the system you are installing on does not have internet
access activation options.
--------------------------------------------------------------------------------
1. Use existing license [default]
2. I want to activate my product using a serial number
3. I want to evaluate my product or activate later
4. I want to activate either remotely, or by using a license file, or by using a
license manager

h. Help
b. Back to the previous menu
q. Quit
--------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]: 2
Please type your serial number (the format is XXXX-XXXXXXXX):
```

Intel编译器安装 VI

```
Trial activation completed successfully.
------------------------------------------------------------------------------
Press "Enter" key to continue:
```

回车

Intel编译器安装 VII

- 是否参加促进计划：2回车

```
Step 4 of 7 | Intel(R) Software Improvement Program
------------------------------------------------------------------------------
Help improve your experience with Intel(R) software
Participate in the design of future Intel software. Select 'Yes' to give us
permission to learn about how you use your Intel software and we will do the
rest.
    - No personally identifiable information is collected
    - There are no additional follow-up emails by opting in
    - You can stop participating at any time

    Learn more about the Intel(R) Software Improvement Program
    http://software.intel.com/en-us/articles/software-improvement-program

------------------------------------------------------------------------------
1. Yes, I am willing to participate and improve Intel software. (Recommended)
2. No, I don't want to participate in the Intel(R) Software Improvement Program
at this time.

b. Back to the previous menu
q. Quit
------------------------------------------------------------------------------
Please type a selection: 2
```

Intel编译器安装 VIII

- 定制：回车

```
Step 5 of 7 | Options
-------------------------------------------------------------------------------
This product can be installed on cluster nodes.
-------------------------------------------------------------------------------
1. Finish configuring cluster [default]

2. Configuration type                                  [ Current node ]

h. Help
b. Back to the previous menu
q. Quit
-------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]:
```

Intel编译器安装 IX

- 定制安装前概要：回车

```
Step 5 of 7 | Options > Pre-install Summary
------------------------------------------------------------------------------
Install location:
    /opt/intel/icsxe/2013.1.046

Component(s) selected:
    Intel(R) MPI Library, Development Kit 4.1 Update 3                  540MB
        Intel(R) MPI Library, Runtime Environment
        Intel(R) MPI Library, Runtime Environment for applications running on
......
   Intel(R) Integrated Performance Primitives 8.1                       2.9GB
        Intel IPP single-threaded libraries

--More--[Press space to continue, 'q' to quit.]
```

Intel编译器安装 X

- 定制：2回车

```
1. Start installation Now [default]
2. Customize installation

h. Help
b. Back to the previous menu
q. Quit
--------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]:
```

Intel编译器安装 XI

- 架构选择：回车

```
Step 5 of 7 | Options > Architecture selection
------------------------------------------------------------------------------
Select the architecture(s) where your applications will run. If unsure, accept
the default options below or see
http://software.intel.com/en-us/articles/about-target-architecture-selection-dur
ing-installation for more information.

Target Architecture(s) of your applications:
------------------------------------------------------------------------------
1. [x]    IA-32
2. [x]    Intel(R) 64

3. Finish architecture selection [default]

Note: This system is an Intel(R) 64 architecture system.
Your application may be built to run on either IA-32 or Intel(R) 64
architectures.

b. Back to the previous menu
q. Quit
------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [3]:
```

Intel编译器安装 XII

- 选项：3回车

```
Step 5 of 7 | Options
-------------------------------------------------------------------------------
You are now ready to begin installation. You can use all default installation
settings by simply choosing the "Start installation Now" option or you can
customize these settings by selecting any of the change options given below
first. You can view a summary of the settings by selecting "Show pre-install
summary".
-------------------------------------------------------------------------------
1. Start installation Now [default]

2. Change install directory       [ /opt/intel/icsxe/2013.1.046 ]
3. Change components to install  [ Custom ]
4. Change advanced options
5. Show pre-install summary

h. Help
b. Back to the previous menu
q. Quit
-------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]:3
```

Intel编译器安装 XIII

- 组件选择：如11回车

```
Step 5 of 7 | Options > Component selection
--------------------------------------------------------------------------------
Select the component you wish to install. When you have completed your changes,
select option 1 to continue with the installation.
--------------------------------------------------------------------------------
1. Finish component selection [default]
2. Intel(R) MPI Library, Development Kit 4.1 Update 3 [All]
3. Intel(R) Trace Analyzer and Collector 8.1 Update 4 [All]
4. Intel(R) VTune(TM) Amplifier XE 2013 Update 15 [All]
5. Intel(R) Inspector XE 2013 Update 9 [All]
6. Intel(R) Advisor XE 2013 Update 5 [All]
7. Intel(R) Fortran Compiler XE 14.0 Update 2 [All]
8. Intel(R) C++ Compiler XE 14.0 Update 2 [All]
9. Intel(R) Debugger 13.0 [All]
10. Intel(R) Math Kernel Library 11.1 Update 2 [Custom]
13. GNU* GDB 7.5 [Custom]
14. Intel(R) MPI Benchmarks 3.2.4 [All]
Install Space Required:      8.1GB
h. Help
b. Back to the previous menu
q. Quit
--------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]:
```

Intel编译器安装 XIV

- 组件选择：如2回车

```
Step 5 of 7 | Options > Component selection
-------------------------------------------------------------------------------
You may choose not to install some components of this product. Optional
components are shown with an option number of the left. Entering that number
will select/unselect that component for installation. When you have completed
your changes, select option 1 to return to previous menu.
-------------------------------------------------------------------------------
1. Finish component selection [default]

2. [x] Intel(R) Integrated Performance Primitives 8.1
3.     [x] Intel IPP single-threaded libraries
4.     [ ] Intel IPP multi-threaded libraries (deprecated)
5.     [ ] Intel IPP performance test application

Install Space Required:      2.9GB

h. Help
b. Back to the previous menu
q. Quit
-------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]: 2
```

Intel编译器安装 XV

- 组件选择：回车

```
Step 5 of 7 | Options > Component selection
--------------------------------------------------------------------------------
Select the component you wish to install. When you have completed your changes,
select option 1 to continue with the installation.
--------------------------------------------------------------------------------
1. Finish component selection [default]
2. Intel(R) MPI Library, Development Kit 4.1 Update 3 [All]
3. Intel(R) Trace Analyzer and Collector 8.1 Update 4 [None]
4. Intel(R) VTune(TM) Amplifier XE 2013 Update 15 [All]
5. Intel(R) Inspector XE 2013 Update 9 [None]
6. Intel(R) Advisor XE 2013 Update 5 [None]
7. Intel(R) Fortran Compiler XE 14.0 Update 2 [All]
8. Intel(R) C++ Compiler XE 14.0 Update 2 [All]
9. Intel(R) Debugger 13.0 [All]
10. Intel(R) Math Kernel Library 11.1 Update 2 [Custom]
13. GNU* GDB 7.5 [Custom]
14. Intel(R) MPI Benchmarks 3.2.4 [None]
Install Space Required:      3.9GB
h. Help
b. Back to the previous menu
q. Quit
--------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]:
```

Intel编译器安装 XVI

- 选项：回车

```
Step 5 of 7 | Options
-------------------------------------------------------------------------------
You are now ready to begin installation. You can use all default installation
settings by simply choosing the "Start installation Now" option or you can
customize these settings by selecting any of the change options given below
first. You can view a summary of the settings by selecting "Show pre-install
summary".
-------------------------------------------------------------------------------
1. Start installation Now [default]

2. Change install directory       [ /opt/intel/icsxe/2013.1.046 ]
3. Change components to install  [ Custom ]
4. Change advanced options
5. Show pre-install summary

h. Help
b. Back to the previous menu
q. Quit
-------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]:
```

- 安装开源组件：回车

```
Installation of Open Source Components
------------------------------------------------------------------------------
Open source components provided under GNU General Public License v3 or Eclipse
Public License v.1.0 will be installed.

Includes:
    GNU* GDB 7.5 (Provided under GNU General Public License v3)
    GDB Eclipse* Integration (Provided under Eclipse Public License v.1.0)

For further details, please refer to the product Release Notes.
------------------------------------------------------------------------------
1. Continue the installation [default]

b. Back to the previous menu
q. Quit
------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]:
```

Intel编译器安装 XVIII

- 不满足的选项要求:

```
Step 5 of 7 | Options > Missing Optional Prerequisite(s)
--------------------------------------------------------------------------------
There are one or more optional unresolved issues. It is highly recommended to
resolve them all before you continue the installation. You can fix them without
exiting from the installation and re-check. Or you can quit from the
installation, fix them and run the installation again.
--------------------------------------------------------------------------------
Missing optional prerequisites
-- 32-bit libraries not found
--------------------------------------------------------------------------------
1. Skip missing optional prerequisites [default]
2. Show the detailed info about issue(s)
3. Re-check the prerequisites
h. Help
b. Back to the previous menu
q. Quit
--------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]:
```

  - 选择2回车，查看详细信息，决定是否做处理，之后可选择3重新检查
  - 1回车

- 开始安装：回车

```
Step 6 of 7 | Installation
-------------------------------------------------------------------------------
Each component will be installed individually. If you cancel the installation,
some components might remain on your system. This installation may take several
minutes, depending on your system and the options you selected.
-------------------------------------------------------------------------------
Installing Intel(R) MPI Library, Runtime Environment for applications running on
IA-32 Architecture component... done
-------------------------------------------------------------------------------
Installing Intel(R) MPI Library, Runtime Environment for applications running on
Intel(R) 64 Architecture component... done
Finalizing product configuration...
-------------------------------------------------------------------------------
Preparing driver configuration scripts... done
-------------------------------------------------------------------------------
Installing drivers. It may take several minutes... done
-------------------------------------------------------------------------------
Sampling driver built successfully
Sampling driver loaded successfully
Sampling driver boot script installed successfully
-------------------------------------------------------------------------------
Press "Enter" key to continue
```

Intel编译器安装 XX

- 结束：回车

```
Step 7 of 7 | Complete
-------------------------------------------------------------------------------
Thank you for installing and for using the Intel(R) Cluster Studio XE 2013 SP1
Update 1 for Linux* OS.

Support services start from the time you install or activate your product. If
you have not already done so, please create your support account now to take
full advantage of your product purchase.

Your support account gives you access to free product updates and upgrades as
well as interactive technical support at Intel(R) Premier Support.

To create your support account, please visit the Intel(R) Software Development
Products Registration Center web site
https://registrationcenter.intel.com/RegCenter/registerexpress.aspx?media=VJ2
-------------------------------------------------------------------------------
q. Quit
-------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [q]:
```

Intel编译器安装 XXI

- 在*~/.bashrc*中设置环境变量：

```
. /opt/intel/composerxe/bin/compilervars.sh intel64
. /opt/intel/impi/4.1.3.048/bin64/mpivars.sh intel64
```

  - 第一行：设置Intel C/C++ Fortran环境
  - 第二行：设置Intel MPI环境
  - 上述是针对Intel64（EM64T、AMD64、X86_64）架构的，如要用IA32，则需将intel64换成ia32

1 Intel C/C++、Fortran编译器简介
- Intel编译器安装配置
- Intel C/C++编译器用法
- Intel Fortran编译器用法
- Intel调试器用法

2 GNU C/C++、Fortran编译器简介
- GNU C/C++编译器用法
- GNU Fortran编译器用法
- GNU程序调试器用法

3 数学函数库
- Intel MKL数学函数库
- 其余数学统计函数库

4 联系信息

编译命令基本格式

基本格式:

- C: *icc␣[options]␣file1␣[file2␣...]*
- C++: *icpc␣[options]␣file1␣[file2␣...]*

注意:

- *[␣]*表示是其内部的选项可选
- 文件名和选项区分大小写

串行及OpenMP并行程序编译举例

- 将C程序yourprog.c编译为可执行文件yourprog:
  icc␣-o␣yourprog␣yourprog.c
- 将C程序yourprog.c编译为目标文件yourprog.o:
  icc␣-c␣yourprog.c
- 将C程序yourprog.c链接*/opt/lib/liblapack.so*后编译为可执行文件yourprog:
  icc␣-o␣yourprog␣-L/opt/lib␣-llapack␣yourprog.c
- 将C程序yourprog.c静态编译为O3优化的可执行文件yourprog:
  icc␣-O3␣-static␣-o␣yourprog␣yourprog.c
- 将C++程序yourprog.cpp编译为可执行文件yourprog:
  icpc␣-o␣yourprog␣yourprog.cpp
- 将OpenMP指令并行的C程序yourprog-omp.c编译为可执行文件yourprog-omp:
  icc␣-o␣yourprog-omp␣-openmp␣yourprog.c

编译时出错信息格式

```
netlog.c(140): error: identifier "hhh" is undefined
    for(int hhh=domain_cnt+1;hhh>TMP;hhh--){
                             ^

netlog.c(156): error: expected an expression
    for(int i=0;i<32;i++)for(int j=0;j<256;j++)if(ip1[i][j]!=0)fprintf(fin,"202.38.%2d.%3d: %d\n",i
        ^
```

- 源文件名(行数): 错误类型:具体说明
- 源代码，^指示出错位置

环境变量 I

- GCCROOT：gcc库路径，只有在因为添加-gcc-name选项导致找不到gcc库时才需要。
- GXX_INCLUDE：gcc头文件路径。
- GXX_ROOT：gcc库路径。
- LIBRARY_PATH：链接库路径
- LD_LIBRARY_PATH：链接共享库（.so库文件）路径
- OMP_*和KMP_*：OpenMP环境及其扩展变量
  - OMP：OMP_DYNAMIC、OMP_MAX_ACTIVE_LEVELS、OMP_NESTED、OMP_NUM_THREADS、OMP_PROC_BIND、OMP_SCHEDULE、OMP_STACKSIZE、OMP_THREAD_LIMIT、OMP_WAIT_POLICY

环境变量 II

- KMP：KMP_AFFINITY、KMP_ALL_THREADS、KMP_BLOCKTIME、KMP_CPUINFO_FILE、KMP_DETERMINISTIC_REDUCTIONS、KMP_DYNAMIC_MODE、KMP_INHERIT_FP_CONTROL、KMP_LIBRARY、KMP_MONITOR_STACKSIZE、KMP_SETTINGS、KMP_STACKSIZE、KMP_VERSION

- GNU和环境及其扩展变量
  - CPATH：C/C++头文件路径
  - C_INCLUDE_PATH：C头文件路径
  - CPLUS_INCLUDE_PATH：C++头文件路径
  - LIBRARY_PATH：库路径

编译过程

- 编译主要分为如下过程：
  - 预处理(Preprocessing)
  - 语义分析(Semantic parsing)
  - 优化(Optimization)
  - 代码生成(Code generation)
  - 链接(Linking)
- 前四项由编译器处理：icc或icpc
- 最后一项由编译器调用链接器处理：xild
- 编译时如添加了如下选项：
  - -c：编译器只生成目标代码（.o文件），需要再显式调用链接器生成可执行程序。
  - -E和-P：只生成预处理文件（.i文件）。
  - [Q]ipo：使用多文件过程间优化（又名全程序优化），将在链接时进行优化。
  - [Q]prof-gen：使用概要导向优化，将在链接时进行优化。

输入文件后缀与类型的关系

编译器默认将按照输入文件的后缀判断文件类型，编译时也可以用编译选项强制指定。

文件名	解释	编译时动作
filename.c	C源文件	传给编译器
filename.C filename.CC filename.cc filename.cpp filename.cxx	C++源文件	传给编译器
filename.a filename.so	库文件	传递给链接器
filename.i	预处理文件	传递给标准输出
filename.o	目标文件	传递给链接器
filename.s	汇编文件	传递给汇编器

输出文件后缀与类型的关系

编译器默认将输出按照文件类型与后缀相对应。

文件名	解释
filename.i	预处理文件，由-p选项生成
filename.o	目标文件，由-c选项生成
filename.s	汇编文件，由-s选项生成
a.out	默认生成的可执行文件

重要编译选项

- 编译选项对运行速度、兼容性等有影响
- 建议仔细看看编译器手册中关于程序优化的部分，特别是IPO、PGO和HLO部分
- 多加测试，选择适合自己程序的编译选项以提高性能
- 以下仅仅介绍部分重要选项

输出及调试选项

- -c：仅编译成目标文件（.o文件）
- -debug [keyword]：启用或禁止生成调试信息。keyword可为none、full、all、minimal、extended、[no]parallel等
- -g：包含调试信息
- -g0：禁止产生符号调试信息
- -o：指定生成的文件名。
- -traceback：在目标文件中生成额外的信息，在运行出错时回溯到源文件

预处理选项 I

- -B*dir*：指定头文件、库文件和可执行文件的搜索路径
- -Dname[=value]：指定传递给编译器的宏及宏值
- -dD：输出源文件中的#define预处理指令
- -dM：输出源文件中的宏定义。
- -dN：与-dD类似，但只输出源文件中的宏名。
- -gcc、-no-gcc和-gcc-sys：决定特定GNU宏（__GNUC__、__GNUC_MINOR__、__GNUC_PATCHLEVEL__）是否定义
- -icc、-no-icc：决定特定Intel宏（__INTEL_COMPILER）是否定义
- -U*name*：去掉特定宏的定义
- -undef：取消所有预定义宏
- -H：现实头文件顺序，并继续编译
- -I<头文件目录>：指明头文件的搜索路径
- -iprefix prefix：指定在头文件目录的先前搜索的路径

预处理选项 II

- -iquote dir：指定用""而不是<>引用的头文件的优先搜索路径
- -isystem*dir*：指定系统头文件的有限搜索路径
- -nostdinc++：不搜索C++的标准头文件路径，搜索其他标准目录
- -X：从搜索路径中去除标准库的路径

优化选项 I

- -fast：最大化整个程序的速度，相当于：-ipo, -O3, -no-prec-div, -static, 和-xHost。这里是所谓的最大化，还是需要结合程序本身使用合适的选项
- -inline-level=[n]：设置inline层数
- -ip：在单个文件中进行过程间优化(Interprocedural Optimizations-IPO)
- -ipo[n]：在多文件中进行过程间优化，n为可产生的目标文件数，为非负整数
- -mkl[=lib]：调用MKL中的特定库，lib可以为parallel（线程库，只是-mkl而没有=明确指定时，默认为此）、sequential（线性库）和cluster（集群特征库）
- -openmp：编译OpenMP程序，注意：只能在同一个节点的CPU上跑OpenMP程序

优化选项 II

- -O<级别>：设定优化级别，默认为O2，O与O2相同，推荐使用。O3为在O2基础之上增加更激进的优化，比如包含循环和内存读取转换和预取等，但在有些情况下速度反而慢，建议在具有大量浮点计算和大数据处理的循环时的程序使用
- -Od：禁止所有优化
- -Ofast：设置某些激进参数优化程序速度，相当于：-O3 -no-prec-div
- -p：进行概要导向优化(Profile Guided Optimization-PGO)
- -simd和-no-simd：指定是否启用SIMD向量化
- -unroll[n]：循环最大可展开的层数，与性能相关
- -vec和-no-vec：指定是否启用向量化

代码生成选项

- -ax*code*：生成针对Intel处理器的多重特征指定自动派发代码路径。code可以为CORE-AVX2、CORE-AVX-I、AVX、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2等
- -m*code*：指定为特征目标生成。code可以为CORE-AVX2、CORE-AVX-I、AVX、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2等
- -m32和-m64：指定生成为IA32或Intel 64架构的代码
- -march=<CPU架构>：指定针对某种CPU架构的处理器进行优化。默认为pentium4，可以为generic、core-avx2、core-avx-i、corei7-avx、corei7、atom、core2、pentium*等
- -mtune=<CPU架构>：指定针对某种CPU架构的处理器进行优化。默认为generic，可以为generic、core-avx2、core-avx-i、corei7-avx、corei7、atom、core2、pentium*等
- -x*code*：指定针目标是何种处理特征。默认为generic，可以为CORE-AVX2、CORE-AVX-I、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2等

链接选项 I

- -Bdynamic：在运行时动态链接所需要的库（.so文件）
- -Bstatic：静态链接用户生成的库（.a文件）
- -cxxlib：指定是否链接gcc提供的C++运行时库和头文件，-cxxlib[=dir]、-cxxlib-nostd和-no-cxxlib
- -L<库目录>：指明库的搜索路径
- -l<库文件>：指明需链接的库名，如库名为libxyz.a，则可用-lxyz指定
- -pthread：使用POSIX线程库支持多线程库
- -shared：产生共享目标而不是可执行文件，必须在编译每个目标文件时使用-fpic选项
- -shared-intel：动态链接Intel库
- -shared-libgcc：动态链接GNU libgcc库，可允许用户越过添加静态链接选项-static时的静态链接限制

链接选项 II

- -static：静态链接所有库
- -static-intel：静态链接Intel库
- -static-libgcc：动态链接GNU libgcc库，可允许用户越过添加动态链接选项-shared时的动态链接限制
- -static-libstdc++：动态链接GNU libstdc++标准库，可允许用户越过添加动态链接选项-shared时的动态链接限制
- -Wl,optlist：传递以,分割的链接参数给链接器
- -Xlinker val：传递以val变量（如链接参数、目标或库）直接给链接器

语言选项

- -ansi：符合与gcc的ansi标准
- -check [keyword[, keyword...]]和-nocheck：是否对某些条件进行检查，keyword可以为：none、[no]arg_temp_created、[no]assume、[no]bounds、[no]format、[no]output_conversion、[no]pointers、[no]stack、[no]uninit、all
- -std=<标准>：标准可以为c89、c99、c9x、gnu89、gnu99、gnu++98、c++0x、c++11、gnu++0x，分别对应不同的标准。默认为-std=gnu89（C程序）和-std=gnu++98（C++程序）
- -stdlib[=keyword]：指定链接所用的C++库。可以为libstdc++（GNU libstdc++库）和libc++（libc++库）
- -strict-ansi：实现严格的ANSI兼容性
- -x <类型>：类型可为c、c++、c-header、cpp-output、c++-cpp-output、assembler、assembler-with-cpp或none，分别表示c源文件等，以使所有源文件都被认为是此类型的

数据选项

- -align和-noalign：数据是否自然对齐
- -fpic、-fPIC和-fno-pic：是否生成位置无关代码。当生成共享代码时，必须添加-fpic
- -fpie、-fPIE：与-fpic类似，生成位置无关代码。不同之处在于-fpie生成的代码只能被链入执行程序
- -mcmodel=*mem_model*：设定内存模型。mem_model可为：
  - small：限制代码和数据在开始的2GB地址空间，默认
  - medium：限制代码在开始的2GB空间，存储数据空间不受此限制
  - large：对于代码和数据存储空间都无限制
- -check-pointers=keyword：是否检查使用指针访问内存时的数组边界。keyword可以为none、rw和write

其它选项

- –help：显示帮助信息
- -v：显示详细编译过程以及编译参数等
- -V：显示编译器版本号
- -w：编译时不显示任何警告，只显示错误
- -wall：编译时显示所有警告

1 Intel C/C++、Fortran编译器简介
- Intel编译器安装配置
- Intel C/C++编译器用法
- Intel Fortran编译器用法
- Intel调试器用法

2 GNU C/C++、Fortran编译器简介
- GNU C/C++编译器用法
- GNU Fortran编译器用法
- GNU程序调试器用法

3 数学函数库
- Intel MKL数学函数库
- 其余数学统计函数库

4 联系信息

支持Fortran标准

- Fortran标准：FORTRAN IV（即66）、77、90、95、2003和2008
- Intel Fortran编译器
  - 支持过时的和已经删除的Fortran特征
  - 完全支持Fortran 95(ANSI X3J3/96-007)和90(ANSI X3.198-1992)
  - 支持大多数Fortran 2003标准(ISO/IEC 1539-1:2004)
  - 开始支持Fortran 2008标准(ISO/IEC 1539-1:2010)的一些特征
  - 支持一些Fortran 2003标准的扩展（官方手册中用绿色表示）

编译命令基本格式

基本格式：

- Fortran：　*ifort ␣[options]␣ file1 ␣[file2 ␣ ...]*

注意：

- *[␣]*表示是其内部的选项可选
- 文件名和选项区分大小写

输入文件后缀与类型的关系

编译器默认将按照输入文件的后缀判断文件类型，编译时也可以用编译选项强制指定。

文件名	解释	动作
filename.a	目标库文件	传给编译器
filename.f filename.for filename.ftn filename.i	固定格式的Fortran源文件	被Fortran编译器编译
filename.fpp filename.FPP filename.F filename.FOR filename.FTN	固定格式的Fortran源文件	自动被Fortran编译器预处理后再被编译
filename.f90 filename.i90	自由格式的Fortran源文件	被Fortran编译器编译
filename.F90	自由格式的Fortran源文件	自动被Fortran编译器预处理后再被编译
filename.s	汇编文件	传递给汇编器
filename.so	库文件	传递给链接器
filename.o	目标文件	传递给链接器

输出文件的后缀与类型的关系

编译器默认将输出按照文件类型与后缀相对应。

文件名	解释	生成方式
filename.o	目标文件	编译时添加-c选项生成
filename.so	共享库文件	编译时指定为共享型，如添加-shared，并不含-c
filename.mod	模块文件	编译含有MODULE声明时的源文件生成
filename.s	汇编文件	编译时添加-S选项生成
a.out	默认生成的可执行文件	编译时没有指定-c时生成

环境变量 I

- GCCROOT：gcc库路径，只有在因为添加-gcc-name选项导致找不到gcc库时才需要
- GXX_INCLUDE：gcc头文件路径
- GXX_ROOT：gcc库路径
- LIBRARY_PATH：链接库路径
- LD_LIBRARY_PATH：链接共享库（.so库文件）路径
- OMP_*和KMP_*：OpenMP环境及其扩展变量
  - OMP：OMP_DYNAMIC、OMP_MAX_ACTIVE_LEVELS、OMP_NESTED、OMP_NUM_THREADS、OMP_PROC_BIND、OMP_SCHEDULE、OMP_STACKSIZE、OMP_THREAD_LIMIT、OMP_WAIT_POLICY

环境变量 II

- KMP：KMP_AFFINITY、KMP_ALL_THREADS、KMP_BLOCKTIME、KMP_CPUINFO_FILE、KMP_DETERMINISTIC_REDUCTIONS、KMP_DYNAMIC_MODE、KMP_INHERIT_FP_CONTROL、KMP_LIBRARY、KMP_MONITOR_STACKSIZE、KMP_SETTINGS、KMP_STACKSIZE、KMP_VERSION

- GNU和环境及其扩展变量
  - CPATH：头文件和module文件路径
  - LIBRARY_PATH：链接库路径
- 编译器运行时环境变量
  - F_UFMTENDIAN：小端到大端（little-endian-to-big-endian）数据转换时的文件号
  - FORT_CONVERTn：指定需要进行小端大端文件转换的文件号

重要编译选项

- 编译选项对运行速度、兼容性等有影响
- 建议仔细看看编译器手册中关于程序优化的部分，特别是IPO、PGO和HLO部分
- 多加测试，选择适合自己程序的编译选项以提高性能
- 以下仅仅介绍部分重要选项

输出及调试选项

- -c：仅编译成目标文件（.o文件）
- -debug [keyword]：启用或禁止生成调试信息。keyword可为none、full、all、minimal、extended、[no]parallel等
- -g：包含调试信息
- -g0：禁止产生符号调试信息
- -o：指定生成的文件名
- -traceback：在目标文件中生成额外的信息，在运行出错时回溯到源文件

预处理选项 I

- -B*dir*：指定头文件、库文件和可执行文件的搜索路径
- -Dname[=value]：指定传递给编译器的宏及宏值
- -dD：输出源文件中的#define预处理指令
- -dM：输出源文件中的宏定义
- -dN：与-dD类似，但只输出源文件中的宏名
- -gcc、-no-gcc和-gcc-sys：决定特定GNU宏（__GNUC__、__GNUC_MINOR__、__GNUC_PATCHLEVEL__）是否定义
- -icc、-no-icc：决定特定Intel宏（__INTEL_COMPILER）是否定义
- -fpp和-nofpp：是否对源代码进行预处理
- -U*name*：去掉特定宏的定义
- -undef：取消所有预定义宏
- -H：现实头文件顺序，并继续编译
- -I<头文件目录>：指明头文件的搜索路径

预处理选项 II

- -iprefix prefix：指定在头文件目录的先前搜索的路径
- -iquote dir：指定用""而不是<>引用的头文件的优先搜索路径
- -isystem*dir*：指定系统头文件的有限搜索路径
- -nostdinc++：不搜索C++的标准头文件路径，搜索其他标准目录
- -X：从搜索路径中去除标准库的路径

优化选项 I

- -fast：最大化整个程序的速度，相当于：-ipo, -O3, -no-prec-div, -static, 和-xHost。这里是所谓的最大化，还是需要结合程序本身使用合适的选项
- -inline-level=[n]：设置inline层数
- -ip：在单个文件中进行过程间优化(Interprocedural Optimizations-IPO)
- -ipo[n]：在多文件中进行过程间优化，n为可产生的目标文件数，为非负整数
- -mkl[=lib]：调用MKL中的特定库，lib可以为parallel（线程库，只是-mkl而没有=明确指定时，默认为此）、sequential（线性库）和cluster（集群特征库）
- -openmp：编译OpenMP程序，注意：只能在同一个节点的CPU上跑OpenMP程序

优化选项 II

- -O<级别>：设定优化级别，默认为O2，O与O2相同，推荐使用。O3为在O2基础之上增加更激进的优化，比如包含循环和内存读取转换和预取等，但在有些情况下速度反而慢，建议在具有大量浮点计算和大数据处理的循环时的程序使用
- -Od：禁止所有优化
- -Ofast：设置某些激进参数优化程序速度，相当于：-O3 -no-prec-div
- -p：进行概要导向优化(Profile Guided Optimization-PGO)
- -simd和-no-simd：指定是否启用SIMD向量化
- -vec和-no-vec：指定是否启用向量化
- -unroll[n]：循环最大可展开的层数，与性能相关

代码生成选项

- -ax*code*：生成针对Intel处理器的多重特征指定自动派发代码路径。code可以为CORE-AVX2、CORE-AVX-I、AVX、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2等
- -m*code*：指定为特征目标生成。code可以为CORE-AVX2、CORE-AVX-I、AVX、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2等
- -m32和-m64：指定生成为IA32或Intel 64架构的代码
- -march=<CPU架构>：指定针对某种CPU架构的处理器进行优化。默认为pentium4，可以为generic、core-avx2、core-avx-i、corei7-avx、corei7、atom、core2、pentium*等
- -mtune=<CPU架构>：指定针对某种CPU架构的处理器进行优化。默认为generic，可以为generic、core-avx2、core-avx-i、corei7-avx、corei7、atom、core2、pentium*等
- -x*code*：指定针目标是何种处理特征。默认为generic，可以为CORE-AVX2、CORE-AVX-I、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2等

链接选项 I

- -Bdynamic：在运行时动态链接所需要的库（.so文件）
- -Bstatic：静态链接用户生成的库（.a文件）
- -cxxlib：指定是否链接gcc提供的C++运行时库和头文件，-cxxlib[=dir]、-cxxlib-nostd和-no-cxxlib
- -L<库目录>：指明库的搜索路径
- -l<库文件>：指明需链接的库名，如库名为libxyz.a，则可用-lxyz指定
- -pthread：使用POSIX线程库支持多线程库
- -shared：产生共享目标而不是可执行文件，必须在编译每个目标文件时使用-fpic选项
- -shared-intel：动态链接Intel库
- -shared-libgcc：动态链接GNU libgcc库，可允许用户越过添加静态链接选项-static时的静态链接限制

链接选项 II

- -static：静态链接所有库
- -static-intel：静态链接Intel库
- -static-libgcc：动态链接GNU libgcc库，可允许用户越过添加动态链接选项-shared时的动态链接限制
- -static-libstdc++：动态链接GNU libstdc++标准库，可允许用户越过添加动态链接选项-shared时的动态链接限制
- -Wl,optlist：传递以,分割的链接参数给链接器
- -Xlinker val：传递以val变量（如链接参数、目标或库）直接给链接器

语言选项 I

- -allow keyword：指明编译器是否运行某些行为，keyword可以为：[no]fpp_comments
- -altparam和-noaltparam：指明编译器是否允许在PARAMETER声明中不使用括号的代替语法
- -assume keyword[, keyword...]：指明采用某些默认。keyword可以为：[no]underscore、[no]2underscores等20多种
- -check [keyword[, keyword...]]和-nocheck：是否对某些条件进行检查，keyword可以为：none、[no]arg_temp_created、[no]assume、[no]bounds、[no]format、[no]output_conversion、[no]pointers、[no]stack、[no]uninit、all
- -extend-source[size]：指明固定格式的Fortran源代码宽度，size可为72、80和132。也可直接用-72、-80和-132指定，默认为72字符
- -implicitnone：指明默认变量名为未定义，建议在写程序时添加implicit none语句，以避免出现由于默认类型造成的错误

语言选项 II

- -fixed：指明Fortran源代码为固定格式，默认由文件后缀决定类别
- -free：指明Fortran源程序为自由格式，默认由文件后缀决定类别
- -names keyword：指明源码中标识符和外部名如何翻译，keyword可以为lowercase、uppercase和as_is
- -nofree：指明Fortran源程序为固定格式
- -pad-source和-nopad-source：指明对固定格式源代码是否在行后部补齐空白
- -stand <标准>和-std<标准>：以指定的Fortran标准进行编译，编译时显示源文件中不符合此标准的信息。标准可为f08、f03、f95、f90和none，分别对应显示不符合Fortran 2008、2003、95、90的代码信息和不显示任何非标准的代码信息，也可写为-std<标准>，此时标准不带f，可为08、03、90、95

语言选项 III

- -us：编译时给外部用户定义的函数名添加一个下划线，等价于-assume underscore，如果编译时显示_函数找不到时也许添加此选项即可解决
- -stdlib[=keyword]：指定链接所用的C++库。可以为libstdc++（GNU libstdc++库）和libc++（libc++库）
- -strict-ansi：实现严格的ANSI兼容性
- -x <类型>：类型可以为c、c++、c-header、cpp-output、c++-cpp-output、assembler、assembler-with-cpp或none，分别表示c源文件等，以使所有源文件都被认为是此类型的

数据选项 I

- -align和-noalign：数据是否自然对齐
- -auto和-noauto：指明是否所有本地、non-SAVEd的变量分配到运行时堆栈。默认为-auto-scalar
- -auto-scalar：指明用INTEGER、REAL、COMPLEX、LOGICAL声明时不具有SAVE属性的变量分配到运行时堆栈
- -save：指明变量存储在静态内存中
- -convert [关键字]：转换无格式数据的类型，比如关键字
  为big_endian和little_endian时，分别表示无格式的输入输出
  为big_endian和little_endian格式，更多格式类型请看编译器手册
- -check-pointers=keyword：是否检查使用指针访问内存时的数组边界。keyword可以为none、rw和write
- -dyncom ”common1,common2,...”：指明在运行时动态分配common块
- -fcommon和-fno-common：指明common块是否为全局定义

数据选项 II

- -fpic、-fPIC和-fno-pic：是否生成位置无关代码。当生成共享代码时，必须添加-fpic
- -fpie、-fPIE：与-fpic类似，生成位置无关代码。不同之处在于-fpie生成的代码只能被链入执行程序
- -intconstant和-nointconstant：指明是否用FORTRAN 77语义决定整数的kind参数
- -double-size size：指明对DOUBLE PRECISION和DOUBLE COMPLEX数据类型的默认KIND。可能值为64(对应real(KIND=8))或128areal((KIND=16))
- -integer-size size：指明整型的默认KIND。size可为16、32或64
- -real-size size：指明实型的默认KIND。size可为32、64或128
- -mcmodel=*mem_model*：设定内存模型。mem_model可为：
  - small：限制代码和数据在开始的2GB地址空间，默认
  - medium：限制代码在开始的2GB空间，存储数据空间不受此限制
  - large：对于代码和数据存储空间都无限制

数据选项 III

- -zero和-nozero： 指明INTEGER、REAL、COMPLEX或LOGICAL声明的本地变量是否初始为0

其它选项

- –help：显示帮助信息
- -v：显示详细编译过程以及编译参数等
- -V：显示编译器版本号
- -w：编译时不显示任何警告，只显示错误
- -wall：编译时显示所有警告

串行及OpenMP并行程序编译举例

- 将Fortran 77程序yourprog.for编译为可执行文件yourprog:
  ifort␣−o␣yourprog␣yourprog.for
- 将Fortran 77程序yourprog.for编译为目标文件yourprog.o:
  ifort␣−c␣yourprog.for
- 将使用lapack库的Fortran 90程序yourprog.f90编译为可执行文件yourprog:
  ifort␣−o␣yourprog␣−L/opt/lib␣−llapack␣yourprog.f90
- 将Fortran 90程序yourprog.f90编译为目标文件yourprog.o:
  ifort␣−c␣yourprog.f90
- 将Fortran 90程序yourprog.f90静态编译为O3优化的可执行文件yourprog:
  ifort␣−O3␣−static␣−o␣yourprog␣yourprog.f90
- 将Fortran 90程序yourprog.f90静态编译为可执行文件yourprog:
  ifort␣−o␣yourprog␣− static␣yourprog.f90
- 将OpenMP指令并行的Fortran 90程序yourprog-omp.f90编译为可执行文件yourprog-omp:
  ifort␣−o␣yourprog−omp␣−openmp␣yourprog.f90

编译时出错信息格式

```
NOlihm.f90(146): error #6404: This name does not have a type, and must have an explicit type.
[NPR]
   n2nd=0;   npr=0
-----------^
NOlihm.f90(542): remark #8290: Recommended relationship between field width 'W' and the number of
6060 format(/i2,'-th layer',i2,'-th element: z=',i3,' a=',f9.5/' Ef=',f7.5,' (keV)'/' c =',f5.3,'
(at%)')
-----------------------------------------------------------------------^
```

- 源文件名(行数): 错误类型:具体说明
- 源代码，^指示出错位置

运行时错误

根据运行时错误代码在官方手册中查找对应错误解释：

- 逐步链接：
  Intel®Fortran Compiler XE 13.0 User and Reference Guides->
  Compiler Reference->
  Error Handling->
  Handling Run-Time Errors->
  List of Run-Time Error Messages
- 直接链接：
  http://scc.ustc.edu.cn/zlsc/chinagrid/intel/compiler_f/main_for/GUID-44448B78-2B87-4998-9828-C8BAEB9F5C9A.htm

1 Intel C/C++、Fortran编译器简介
- Intel编译器安装配置
- Intel C/C++编译器用法
- Intel Fortran编译器用法
- Intel调试器用法

2 GNU C/C++、Fortran编译器简介
- GNU C/C++编译器用法
- GNU Fortran编译器用法
- GNU程序调试器用法

3 数学函数库
- Intel MKL数学函数库
- 其余数学统计函数库

4 联系信息

串行程序调试：idb

- 利用Intel编译器自带调试器：命令行的*idbc*和图形界面的*idb*，且可以调试OpenMP和MPI并行程序，进入后运行help可以查看具体命令
- 编译时需要添加*−g*参数，比如：

  ifort␣−g␣−o␣yourprog␣yourprog.f90
- 启动调试：
  - 无初始调试命令文件方式：*idbc␣./yourprog*
  - 有初始调试命令文件方式：*idbc␣−command␣dbg.txt␣./yourprog*

  dbg.txt文件存储调试命令，每行一条命令，启动时自动按顺序执行，其内容类似：

```
break 139
run
```

详细用法，参见：

- ChinaGrid高性能计算集群使用指南
- Intel®Debugger User's and Reference Guide

1 Intel C/C++、Fortran编译器简介

2 GNU C/C++、Fortran编译器简介

3 数学函数库

4 联系信息

GNU C/C++、Fortran编译器简介

GNU C/C++ Fortran编译器（GCC）为系统自带的默认编译器，用户无需特殊设置即可使用

- 编译C、C++源程序的命令：分别为gcc和g++
- 编译Fortran 77和9x[2]、200x[3]源程序的命令：分别为g77和gfortran：
  - gfortran：属于GCC 4系列，可以编译Fortran 77及9x、200x源程序
  - g77：属于GCC 3.4系列，不可编译Fortran 9x、200x源程序

[2]指的为Fortran 90、95标准
[3]指的是Fortran 2003和2008标准

1 Intel C/C++、Fortran编译器简介
- Intel编译器安装配置
- Intel C/C++编译器用法
- Intel Fortran编译器用法
- Intel调试器用法

2 GNU C/C++、Fortran编译器简介
- GNU C/C++编译器用法
- GNU Fortran编译器用法
- GNU程序调试器用法

3 数学函数库
- Intel MKL数学函数库
- 其余数学统计函数库

4 联系信息

编译命令基本格式

基本格式：

- C：*gcc␣[options]␣file1␣[file2␣...]*
- C++：*g++␣[options]␣file1␣[file2␣...]*

注意：

- *[␣]*表示是其内部的选项可选
- 文件名和选项区分大小写

输入文件后缀与类型的关系

编译器默认将按照输入文件的后缀判断文件类型，编译时也可以用编译选项强制指定。

文件名	解释	动作
filename.c	C源文件	传给编译器
filename.C filename.CC filename.cc filename.cpp filename.cxx	C++源文件	传给编译器
filename.a filename.so	库文件	传递给链接器
filename.i	已预处理的文件	传递给标准输出
filename.o	目标文件	传递给链接器
filename.s	汇编文件	传递给汇编器

输出文件后缀与类型的关系

编译器默认将输出按照文件类型与后缀相对应。

文件名	解释
filename.i	已预处理的文件，由使用-p选项生成
filename.o	目标文件，由添加-c选项生成
filename.s	汇编文件，由添加-s选项生成
a.out	默认生成的可执行文件

重要编译选项

- 编译选项对运行速度、编译的兼容性等有影响
- 建议仔细看看编译器手册，多加测试，选择适合自己程序的编译选项以提高性能
- 以下仅仅介绍部分重要选项

控制文件类型的选项 I

- -x language：明确指定而非让编译器判断输入文件的类型。language可为：
  - c、c-header、c-cpp-output
  - c++、c++-header、c++-cpp-output
  - objective-c、objective-c-header、objective-c-cpp-output
  - objective-c++ objective-c++-header objective-c++-cpp-output
  - assembler、assembler-with-cpp
  - ada
  - f95、f95-cpp-input
  - java
  - treelang

  当language为none时，禁止任何明确指定的类型，其类型由文件名后缀决定
- -c：仅编译成目标文件（.o文件），并不进行链接
- -o file：指定生成的文件名

控制文件类型的选项 II

- -v：详细模式，显示在每个命令执行前显示其命令行
- -###：显示编译器、汇编器、链接器的调用信息但并不进行实际编译，在脚本中可以用于捕获驱动器生成的命令行
- –help：显示帮助信息
- –target-help：显示目标平台的帮助信息
- –version：显示编译器版本信息

C语言选项

- -ansi：C模式时，支持所有ISO C90指令。在C++模式时，去除与ISO C++冲突的GNU扩展
- -std=standard：控制语言标准，standard可为c89、iso9899:1990、iso9899:199409、c99、c9x、iso9899:1999、iso9899:199x、gnu89、gnu99、gnu9x、c++98、gnu++98

警告选项

- -fsyntax-only：仅仅检查代码的语法错误，并不进行其它操作
- -w：编译时不显示任何警告，只显示错误
- -Wfatal-errors：遇到第一个错误就停止，而不尝试继续运行显示更多错误信息

调试选项

- -g：包含调试信息
- -ggdb：包含利用gbd调试时所需要的信息

优化选项

- -O[level]：设置优化级别。优化级别level可以设置为0、1、2、3、s，默认为-O0
- -Ofast：设置优化级别，相当于-O3

预处理选项

- -C：预处理时保留C源文件中的注释
- -D name：预处理时定义宏name的值为1
- -D name=def：预处理时定义name为def
- -U name：预处理时去除的任何name初始定义
- -undef：不预定义系统或GCC声明的宏，但标准预定义的宏仍被定义
- -dD：显示源文件中定义的宏及其值到标准输出
- -dI：显示预处理中包含的所有文件，包括文件名和定义时的行号信息
- -dM：显示预处理时源文件中定义的宏及其值，含定义时文件名和行号
- -dN：与-dD类似，但只显示源文件中定义的宏，而不显示宏值
- -E：预处理.c文件，将结果发给标准输出，不进行编译、汇编或链接
- -I<头文件目录>：指明头文件的搜索路径
- -M：打印make的依赖关系到标准输出
- -MD：打印make的依赖关系到文件file.d，file是编译文件的根名字
- -MM：打印make的依赖关系到标准输出，但忽略系统头文件
- -MMD：打印make的依赖关系到文件file.d，其中file是编译的文件的根名字，但忽略系统头文件
- -P：预处理每个文件，并保留每个file.c文件预处理后的结果到file.i

链接选项

- -pie：在支持的目标上生成位置无关的可执行文件
- -s：从可执行文件中去除所有符号表
- -rdynamic：添加所有符号表到动态符号表中
- -static：静态链接所需的库
- -shared：生成共享目标而不是可执行文件，必须在编译每个目标文件时使用-fpic选项
- -shared-libgcc：使用共享libgcc库
- -static-libgcc：使用静态libgcc库
- -u <symbol>：确保symbol未定义，强制链接一个库模块来定义它
- -I<头文件目录>：指明头文件的搜索路径
- -l<库文件>：指明所链接的库名，如库为libxyz.a，可用-lxyz指定
- -L<库目录>：指明库的搜索路径
- -B<路径>：设置寻找可执行文件、库、头文件、数据文件等路径

CPU平台相关选项

- -mtune=cpu-type：设置优化针对的CPU类型，可为：generic、core2、corei7、corei7-avx、core-avx-i、core-avx2、k8、opteron等
- -march=cpu-type：设置指令针对的CPU类型，可为：generic、core2、corei7、corei7-avx、core-avx-i、core-avx2、k8、opteron等
- -mieee-fp和-mno-ieee-fp：浮点操作是否严格按照IEEE标准

约定成俗的选项

- -fpic：生成位置无关的代码以用于共享库
- -fPIC：如果目标机器支持，将生成位置无关的代码
- -fopenmp：编译OpenMP并行程序
- -fpie和-fPIE：与-fpic和-fPIC类似，但生成的位置无关代码只能链接到可执行文件中

串行及OpenMP并行程序编译举例

- 将C程序yourprog.c编译为可执行文件yourprog:
  gcc␣−o␣yourprog␣yourprog.c
- 将C程序yourprog.c编译为目标文件yourprog.o:
  gcc␣−c␣yourprog.c
- 将使用lapack库的C程序yourprog.c编译为可执行文件yourprog:
  gcc␣−o␣yourprog␣−L/opt/lib␣−llapack␣yourprog.c
- 将C程序yourprog.c静态编译为O3优化的可执行文件yourprog:
  gcc␣−O3␣−static␣−o␣yourprog␣yourprog.c
- 将C++程序yourprog.cpp编译为可执行文件yourprog:
  g++␣−o␣yourprog␣yourprog.cpp
- 将OpenMP指令并行的C程序yourprog-omp.c编译为可执行文件yourprog-omp:
  gcc␣−o␣yourprog−omp␣−fopenmp␣yourprog.c

编译时出错信息格式

```
netlog.c: In function 'main':
netlog.c:84:7: error: 'for' loop initial declarations are only allowed in C99 mode
netlog.c:84:7: note: use option -std=c99 or -std=gnu99 to compile your code
```

- 源文件名: 函数中
- 源文件名:行数:列数:错误类型:具体说明
- 源文件名:行数:列数:注解:解决办法

1 Intel C/C++、Fortran编译器简介
- Intel编译器安装配置
- Intel C/C++编译器用法
- Intel Fortran编译器用法
- Intel调试器用法

2 GNU C/C++、Fortran编译器简介
- GNU C/C++编译器用法
- GNU Fortran编译器用法
- GNU程序调试器用法

3 数学函数库
- Intel MKL数学函数库
- 其余数学统计函数库

4 联系信息

编译命令基本格式

基本格式：

- 4.x.y版本编译器：*gfortran␣[options]␣file1␣[file2␣...]*
- 3.x.y版本编译器：*g77␣[options]␣file1␣[file2␣...]*

注意：

- *[␣]*表示是其内部的选项可选
- 文件名和选项区分大小写
- 后续内容为关于4.x.y版本gfortran的，3.x.y版本的g77有所不同

输入文件后缀与类型的关系

编译器默认将按照输入文件的后缀判断文件类型，编译时也可以用编译选项强制指定。

文件名	解释	动作
filename.a	目标库文件	传给编译器
filename.f filename.for filename.ftn filename.i	固定格式的Fortran源文件	被Fortran编译器编译
filename.fpp filename.FPP filename.F filename.FOR filename.FTN	固定格式的Fortran源文件	自动被Fortran编译器预处理后再被编译
filename.f90 filename.f95 filename.f03 filename.f08 filename.i90 filename.i95 filename.i03 filename.i08	自由格式的Fortran源文件	被Fortran编译器编译
filename.F90 filename.F95 filename.F03 filename.F08	自由格式的Fortran源文件	自动被Fortran编译器预处理后再被编译
filename.s	汇编文件	传递给汇编器
filename.so	库文件	传递给链接器
filename.o	目标文件	传递给链接器

输出文件的后缀与类型的关系

编译器默认将输出按照文件类型与后缀相对应。

文件名	解释	生成方式
filename.o	目标文件	编译时添加-c选项生成
filename.so	共享库文件	编译时指定为共享型，如添加-shared，并不含-c
filename.mod	模块文件	编译含有MODULE声明时的源文件生成
filename.s	汇编文件	编译时添加-S选项生成
a.out	默认生成的可执行文件	编译时没有指定-c时生成

重要编译选项

- 编译选项对运行速度、编译的兼容性等有影响
- 建议仔细看看编译器手册，多加测试，选择适合自己程序的编译选项以提高性能
- gfortran支持所有gcc选项
- 以下仅仅介绍部分重要选项

控制Fortran语言类型的选项

- -ffree-form和-ffixed-form：声明源文件是自由格式还是固定格式，默认Fortran 9x、200x的源文件为自由格式，Fortran 77等的源文件为固定格式
- -fdefault-double-8：设置DOUBLE PRECISION类型为8比特
- -fdefault-integer-8：设置INTEGER和LOGICAL类型为8比特
- -fdefault-real-8：设置REAL类型为8比特
- -fno-backslash：将反斜线(\)当作正常字符（非转义符）处理
- -ffixed-line-length-<n>：设置固定格式源代码的行宽为n
- -ffree-line-length-<n>：设置自由格式源代码的行宽为n
- -fmax-identifier-length=<n>：设置名称的最大字符长度为n，Fortran 95和200x的长度分别为31和65
- -fimplicit-none：禁止变量的隐式声明，所有变量需显式声明
- -fcray-pointer：支持Cray指针扩展
- -fopenmp：编译OpenMP并行程序
- -std=<std>：指明Fortran标准，std可以为gnu、f95、f2003、f2008、legacy
- -M<dir>和-J<dir>：指定编译时保存生成的模块文件目录
- -fconvert=<conversion>：指定对无格式Fortran数据文件表示方式，其值可为：native，默认值；swap，在输入输出时从大端（big-endian）到小端（little-endian）交换比特，或相反；big-endian，用大端方式读写；little-endian，用小端方式读写。x86架构为小端，IBM POWER架构为大端

一般选项

- -c：仅编译成目标文件（.o文件），并不进行链接
- -o file：指定生成的文件名
- -v：详细模式，显示在每个命令执行前显示其命令行
- -###：显示编译器、汇编器、链接器的调用信息但并不进行实际编译，在脚本中可以用于捕获驱动器生成的命令行
- –help：显示帮助信息
- –target-help：显示目标平台的帮助信息
- –version：显示编译器版本信息

警告选项

- -fsyntax-only：仅仅检查代码的语法错误，并不进行其余操作
- -w：编译时不显示任何警告，只显示错误
- -Wfatal-errors：遇到第一个错误就停止，而不尝试继续运行

调试选项

- -g：包含调试信息
- -ggdb：包含利用gbd调试时所需要的信息

优化选项

- -O[level]：设置优化级别。优化级别level可以设置为0、1、2、3、s，默认为-O0
- -Ofast：设置优化级别，相当于-O3

预处理选项

- -C：保留预处理的C源文件中的注释
- -D name：在预处理中定义宏name的值为1
- -D name=def：在预处理中定义name为def
- -U name：去除预处理中的任何name初始定义
- -undef：不预定义系统或GCC声明的宏，但标准预定义的宏仍被定义
- -dD：显示源文件中定义的宏及其值到标准输出
- -dI：显示预处理中包含的所有文件，包括文件名和定义时的行号
- -dM：显示预处理时源文件中定义的宏及值，含定义时文件名和行号
- -dN：与-dD类似，但只显示源文件中定义的宏，而不显示宏值
- -E：预处理各文件，将结果发给标准输出，不进行编译、汇编或链接
- -I<头文件目录>：指明头文件的搜索路径
- -M：打印make的依赖关系到标准输出
- -MD：打印make的依赖关系到文件file.d，file是编译文件的根名字
- -MM：打印make的依赖关系到标准输出，但忽略系统包含
- -MMD：打印make的依赖关系到文件file.d，其中file是编译的文件的根名字，但忽略系统头文件
- -P：预处理每个文件，并保留每个file.c文件预处理后的结果到file.i

链接选项

- -pie：在支持的目标上生成位置无关的可执行文件
- -s：从可执行文件中去除所有符号表
- -rdynamic：添加所有符号表到动态符号表中
- -static：静态链接所需的库
- -shared：生成共享目标而不是可执行文件，必须在编译每个目标文件时使用-fpic选项
- -shared-libgcc：使用共享libgcc库
- -static-libgcc：使用静态libgcc库
- -u <symbol>：确保符号symbol未定义，强制连接一个库模块定义它
- -I<头文件目录>：指明头文件的搜索路径
- -l<库文件>：指明需链接的库名，如库为libxyz.a，则可用-lxyz指定
- -L<库目录>：指明库的搜索路径
- -B<路径>：设置寻找可执行文件、库、头文件、数据文件等路径

平台相关选项

- -mtune=cpu-type：设置优化针对的CPU类型，可为：generic、core2、corei7、corei7-avx、core-avx-i、core-avx2、k8、opteron等
- -march=cpu-type：设置指令针对的CPU类型，可为：generic、core2、corei7、corei7-avx、core-avx-i、core-avx2、k8、opteron等
- -mieee-fp和-mno-ieee-fp：浮点操作是否严格按照IEEE标准

约定成俗的选项

- -fno-automatic：将程序单元的本地变量和数组声明具有SAVE属性
- -ff2c：与g77和f2c生成的代码兼容
- -fno-underscoring：不在名字后添加_。注意：gfortran默认行为与g77和f2c不兼容，为了兼容需要加-ff2c选项。除非使用者了解与现有系统环境的集成，否则不建议使用-fno-underscoring选项
- -funderscoring：对外部名字没有_的加_，以与一些Fortran编译器兼容
- -fsecond-underscore：默认gfortran对外部名称添加一个_，如果使用此选项，那么将添加两个_。此选项当使用-fno-underscoring选项时无效。此选项当使用-ff2c时默认启用
- -fpic：生成位置无关的代码以用于共享库
- -fPIC：如果目标机器支持，将生成位置无关的代码
- -fpie和-fPIE：与-fpic和-fPIC类似，但生成的位置无关代码只能链接到可执行文件中

串行及OpenMP并行程序编译举例

- 将Fortran 77程序yourprog.for编译为可执行文件yourprog:
  gfortran␣−o␣yourprog␣yourprog.for
- 将Fortran 90程序yourprog.f90编译为可执行文件yourprog:
  gfortran␣−o␣yourprog␣yourprog.f90
- 将使用lapack库的Fortran 90程序yourprog.f90编译为可执行文件yourprog:
  gfortran␣−o␣yourprog␣−L/opt/lib␣−llapack␣yourprog.f90
- 将Fortran 90程序yourprog.f90编译为目标文件yourprog.o:
  gfortran␣−c␣yourprog.f90
- 将Fortran 90程序yourprog.f90静态编译为O3优化的可执行文件yourprog:
  gfortran␣−O3␣−static␣−o␣yourprog␣yourprog.f90
- 将OpenMP指令并行的Fortran 90程序yourprog-omp.f90编译为可执行文件yourprog-omp:
  gfortran␣−o␣yourprog−omp␣−fopenmp␣yourprog.f90

编译时出错信息格式

```
NOlihm.f90:146.14:

   n2nd=0;    npr=0
                  1
Error: Symbol 'npr' at (1) has no IMPLICIT type
```

- 源文件名:行数:列数:
- 源文件代码
- 1指示出错位置
- 错误类型: 具体说明

1 Intel C/C++、Fortran编译器简介
- Intel编译器安装配置
- Intel C/C++编译器用法
- Intel Fortran编译器用法
- Intel调试器用法

2 GNU C/C++、Fortran编译器简介
- GNU C/C++编译器用法
- GNU Fortran编译器用法
- GNU程序调试器用法

3 数学函数库
- Intel MKL数学函数库
- 其余数学统计函数库

4 联系信息

GNU程序调试器gdb简介

- GNU调试器gdb可用于调试C/C++、Fortran等语言编写的程序
- 程序编译时需要添加-g编译选项
- 基本启动方式:
  - *gdb␣[−help]␣[−nx]␣[−q]␣[−batch]␣[−cd=dir]␣[−f]*
    [−b␣bps]␣[− tty =dev]␣[−s␣ symfile]␣[−e␣prog]␣[−se␣prog]
    [−c␣core]␣[−x␣cmds]␣[−d␣dir]␣[prog␣[core|procID]]
  - *gdb␣[options]␣−−args␣prog␣[arguments]*
- 注意：程序本身含有参数时，前面需要用–args指定，否则不起作用

gdb启动参数

- -b bps：指定远程调试时的线速，波特率或者比特/每秒
- -batch：以批处理模式运行-x FILE（和.gdbinit）中的命令，退出值为0时表示成功运行指定的所有命令，否则表示有错误发生
- -c FILE, -core=FILE：指定使用core文件
- -cd=directory：以directory目录为其工作目录
- -e FILE, -exec=FILE：指定使用的可执行文件
- -h, -help：显示帮助信息
- -se=file：从文件file中读取符号表，并当可执行文件使用
- -q, -quiet：安静模式，启动时不显示版权等信息
- -tty=device：以设备device做为标准输入输出设备
- –args：将可执行文件名后面的参数传递给此可执行文件
- -tui：以带有专门源代码显示窗口的模式运行
- -write：支持写入可执行和core文件
- -x FILE, -command=FILE：指定启动gbd后执行的操作命令

进入gdb后的操作

进入gdb后的命令非常多，这里仅仅解释几个主要的，详细的请进入gdb后利用help命令查看，里面的信息以及命令比man gdb多得多。

- break [file:]function：设置[file中的]名字为function的函数断点
- run [arglist]：[以arglist参数列表]启动程序
- bt：显示程序的堆栈stack
- print expr：打印表达式的值
- c：继续运行因为断点等终止的程序
- next：执行程序中的下一行语句，如此行是函数，将执行完此函数
- edit [file:]function：编辑当前停止行的代码
- list [file:]function：打印当前停止行附近的代码
- step：执行行程序中的下一行语句，如此行是函数将进入函数内
- help [name]：显示[GDB name命令或]一般GDB信息
- quit：退出GDB

1 Intel C/C++、Fortran编译器简介

2 GNU C/C++、Fortran编译器简介

3 数学函数库

4 联系信息

数学函数库简介

- 常见数学（数值）函数的集合
- 正确性有保证
- 程序开发时直接调用即可，避免重复开发
- 针对特定平台做了优化

1 Intel C/C++、Fortran编译器简介
- Intel编译器安装配置
- Intel C/C++编译器用法
- Intel Fortran编译器用法
- Intel调试器用法

2 GNU C/C++、Fortran编译器简介
- GNU C/C++编译器用法
- GNU Fortran编译器用法
- GNU程序调试器用法

3 数学函数库
- Intel MKL数学函数库
- 其余数学统计函数库

4 联系信息

Intel MKL

- Intel核心数学库(Math Kernel Library, MKL)，用户可以直接调用，以提高性能、加快开发。
- 安装后的路径类似下面，具有Intel 64(EM64T、AMD64、X86_64)和IA-32版本：
  - */opt/intel/composer_xe_2013.2.146/mkl*
  - */opt/intel/Compiler/11.1/073/mkl*
  - */opt/intel/composerxe−2011.3.174/mkl*
  - */opt/intel/mkl/10.0.4.023*

环境设置

设置针对Intel 64的MKL所需的INCLUDE、LD_LIBRARY_PATH和MANPATH等环境变量：

- bash下可在*~/.bashrc*之类文件中添加类似以下代码之一（与具体版本对应）

```
../opt/intel/composer_xe_2013.2.146/bin/compilervars.sh intel64
../opt/intel/mkl/10.0.4.023/tools/environment/mklvarsem64t.sh
../opt/intel/Compiler/11.1/073/mkl/tools/environment/mklvarsem64t.sh
../opt/intel/composerxe/mkl/bin/intel64/mklvars_intel64.sh
```

- csh下可在*~/.login*之类文件中添加以下代码之一（与具体版本对应）

```
source /opt/intel/composer_xe_2013.2.146/bin/compilervars.csh intel64
source /opt/intel/mkl/10.0.4.023/tools/environment/mklvarsem64t.csh
source /opt/intel/Compiler/11.1/073/mkl/tools/environment/mklvarsem64t.c
source /opt/intel/composerxe/mkl/bin/intel64/mklvars_intel64.csh
```

MKL主要内容

- BLAS(level 1, 2, and 3)和LAPACK线性代数程序：支持向量、向量-矩阵、矩阵-矩阵操作。
- PARDISO*直接离散算子：一种迭代稀疏矩阵解算器，支持用于求解方程的离散系统的离散BLAS(level 1, 2, and 3)程序。
- ScaLAPACK分布式的线性代数程序：含有基本线性代数通信子程序（Basic Linear Algebra Communications Subprograms, BLACS)和并行基本线性代数子程序（Parallel Basic Linear Algebra Subprograms, PBLAS）。
- 快速傅立叶变换子程序（Fast Fourier transform, FFT）：支持1、2、3维和混合基数（不局限于2的次方），并具有分布式并行的版本。
- 向量数学库（Vector Math Library, VML）：优化后的针对向量的数学操作程序。
- 向量统计库（Vector Statistical Library, VSL）：提供高性能的向量化随机数产生子(RNG)，主要针对几率分布、卷积和相关性程序、摘要统计功能。
- 数据拟合库（Data Fitting Library）：提供基于样条函数逼近能力，导数和积分，和搜索。

MKL目录内容

MKL主目录，比如/opt/intel/composer_xe_2013.2.146/mkl

主目录下的子目录	内容
bin	存储设置环境变量的脚本
bin/ia3	针对IA-32架构
bin/intel64	针对Intel 64架构
benchmarks/linpack	包含OpenMP版的LINPACK的基准程序
benchmarks/mp_linpack	包含MPI版的LINPACK的基准程序
examples	一些例子，建议用户参考学习
include	含有INCLUDE文件
include/ia32	含有针对IA-32架构的Fortran 95 .mod文件
include/intel64/lp64	含有针对Intel 64架构及LP64接口的Fortran 95 .mod文件
include/intel64/ilp64	含有针对Intel 64架构及ILP64接口的Fortran 95 .mod文件
include/fftw	FFTW2和FFTW3接口的头文件
interfaces/blas95	包含BLAS的Fortran 90封装及用于编译成库的makefile
interfaces/fftw2xc	包含2.x版FFTW(C接口)封装及用于编译成库的makefile
interfaces/fftw2xf	包含2.x版FFTW(Fortran接口)封装及用于编译成库的makefile
interfaces/fftw3xc	包含3.x版FFTW(C接口)封装及用于编译成库的makefile
interfaces/fftw3xf	包含3.x版FFTW(Fortran接口)封装及用于编译成库的makefile
interfaces/fftw2x_cdft	包含2.x版MPI FFTW(集群FFT)封装及用于编译成库的makefile
interfaces/lapack95	包含LAPACK的Fortran 90封装及用于编译成库的makefile
lib/32	包含IA-32架构的静态库和共享目标文件
lib/intel64	包含Intel 64架构的静态库和共享目标文件
tests	一些测试文件
tools/builder	包含用于生成定制动态可链接库的工具

文档在上层目录的Documentation/en_US/mkl子目录下，如：/opt/intel/composer_xe_2013.2.146/Documentation/en_US/mkl。

动态库与静态库

- 静态函数库
  - 库名一般是libxxx.a
  - 整个函数库的所有数据都会被整合进目标代码中
  - 优点：
    - 编译后的执行程序不需要外部的函数库支持
    - 执行速度快
  - 缺点：
    - 编译成的文件比较大
    - 如静态函数库改变了，那么程序必须重新编译
    - 如多个应用程序使用的话，会被装载多次，浪费内存
- 动态函数库
  - 库名一般是libxxx.so
  - 编译时没有被编译进目标代码中，执行到相关函数时才调用该函数库里的相应函数
  - 优点
    - 产生的可执行文件比较小
    - 动态函数库的改变并不影响程序，动态函数库的升级比较方便
    - 共享：多个应用程序可以使用同一个动态库，启动多个应用程序的时候，只需要将动态库加载到内存一次即可
  - 缺点：1、程序的运行环境中必须提供相应的库；2、运行速度慢

利用-mkl编译器参数

Intel Composer XE编译器支持采用-mkl[4]参数链接Intel MKL：

- -mkl或-mkl=parallel：采用标准线程（OpenMP）Intel MKL库链接
- -mkl=sequential：采用串行Intel MKL库链接
- -mkl=cluster：采用Intel MPI和串行MKL库链接
- 对Intel 64架构的系统，默认使用LP64接口链接程序

[4]是-mkl，不是-lmkl，其它编译器未必支持此-mkl选项。

使用单一动态库

- 可以通过使用Intel MKL Single Dynamic Library(SDL)来简化链接行。
- 为了使用SDL，请在链接行上添加libmkl_rt.so，如
  icc 」application .c」−lmkl_rt
- SDL使得可以在运行时选择Intel MKL的接口和线程。默认使用SDL链接时提供:
  - 对Intel 64架构的系统，使用LP64接口链接程序
  - Intel线程
- 如需要使用其它接口或改变线程性质，含使用串行版本Intel MKL等，需要使用函数或环境变量来指定选择，参见动态选择接口和线程层部分。

选择所需库进行链接

- 选择所需库进行链接，一般需要：
  - 从接口层(Interface layer)和线程层(Threading layer)各选择一个库
  - 从计算层(Computational layer)和运行时库(run-time libraries, RTL)添加仅需的库
- 链接应用程序时的对应库参见下表

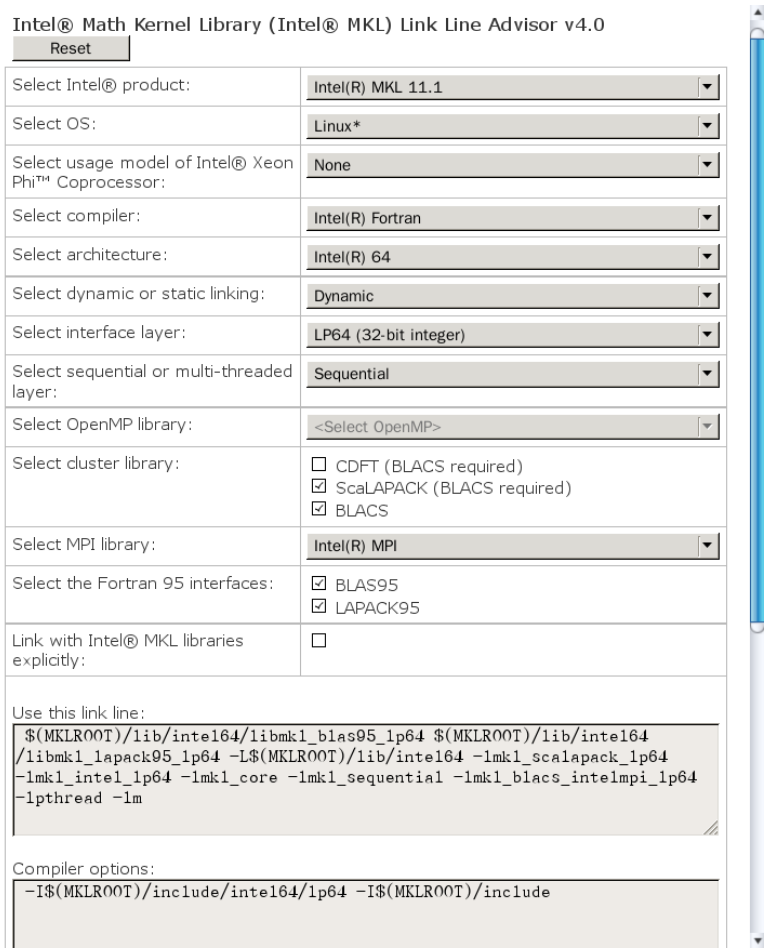
	接口层	线程层	计算层	运行库
IA-32架构，静态链接	libmkl_intel.a	libmkl_intel_thread.a	libmkl_core.a	libiomp5.so
IA-32架构，动态链接	libmkl_intel.so	libmkl_intel_thread.so	libmkl_core.so	libiomp5.so
Intel 64架构，静态链接	libmkl_intel_lp64.a	libmkl_intel_thread.a	libmkl_core.a	libiomp5.so
Intel 64架构，动态链接	libmkl_intel_lp64.so	libmkl_intel_thread.so	libmkl_core.so	libiomp5.so

- SDL会自动链接接口、线程和计算库，简化了链接处理。下表列出的是采用SDL动态链接时的Intel MKL库

	单一动态库	运行时库
IA-32和Intel 64架构	libmkl_rt.so	libiomp5.so

使用链接行顾问

- Intel提供了网页方式的链接行顾问帮用户设置所需的MKL链接参数
- 访问http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor
- 按照提示输入所需要信息即可获取链接Intel MKL时所需要的参数



使用命令行链接工具

- 使用Intel MKL的命令行链接工具可简化使用Intel MKL编译程序
- 本工具不仅可以给出所需的选项、库和环境变量，还可执行编译和生成可执行程序。
- *mkl_link_tool* 命令安装在*<mkl directory>/tools*，主要有三种模式：
  - 查询模式：返回所需的编译器参数、库或环境变量等：
    - 获取Intel MKL库：*mkl_link_tool␣−libs␣[Intel␣MKL␣Link␣Tool␣options]*
    - 获取编译参数：*mkl_link_tool␣−opts␣[Intel␣MKL␣Link␣Tool␣options]*
    - 获取编译环境变量：*mkl_link_tool␣−env␣[Intel␣MKL␣Link␣Tool␣options]*
  - 编译模式：可编译程序：
    - *mkl_link_tool␣[options]␣<compiler>␣[options2]␣file1␣[file2␣...]*
  - 交互模式：采用交互式获取所需要的参数等：
    - *mkl_link_tool␣− interactive*
- 参见：
  http://software.intel.com/en-us/articles/mkl-command-line-link-tool

在Intel 64架构上链接

- 在这些例子中：
  - MKLPATH=$MKLROOT/lib/intel64
  - MKLINCLUDE=$MKLROOT/include
- 如已设置好环境变量，那么：
  - 在所有例子中可以略去-I$MKLINCLUDE
  - 在所有动态链接的例子中可以略去-L$MKLPATH

使用LP64接口的并行Intel MKL库链接

- 静态链接myprog.f:

```
ifort myprog.f -L$MKLPATH -I$MKLINCLUDE \
-Wl,--start-group $MKLPATH/libmkl_intel_lp64.a \
$MKLPATH/libmkl_intel_thread.a $MKLPATH/libmkl_core.a \
-Wl,--end-group -liomp5 -lpthread -lm
```

- 动态链接myprog.f:

```
ifort myprog.f -L$MKLPATH -I$MKLINCLUDE \
-lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5 -lpthread -lm
```

使用LP64接口的串行Intel MKL库链接

- 静态链接myprog.f:

```
ifort myprog.f -L$MKLPATH -I$MKLINCLUDE \
-Wl,--start-group $MKLPATH/libmkl_intel_lp64.a \
$MKLPATH/libmkl_sequential.a $MKLPATH/libmkl_core.a \
-Wl,--end-group -lpthread -lm
```

- 动态链接myprog.f:

```
ifort myprog.f -L$MKLPATH -I$MKLINCLUDE \
-lmkl_intel_lp64 -lmkl_sequential -lmkl_core -lpthread -lm
```

使用ILP64接口的并行Intel MKL库链接

- 动态链接myprog.f:

```
ifort myprog.f -L$MKLPATH -I$MKLINCLUDE \
-Wl,--start-group $MKLPATH/libmkl_intel_ilp64.a \
$MKLPATH/libmkl_intel_thread.a $MKLPATH/libmkl_core.a \
-Wl,--end-group -liomp5 -lpthread -lm
```

- 动态链接myprog.f:

```
ifort myprog.f -L$MKLPATH -I$MKLINCLUDE \
-lmkl_intel_ilp64 -lmkl_intel_thread -lmkl_core -liomp5 -lpthread -lm
```

使用串行或并行（调用函数或设置环境变量选择线程串行模式，并设置接口）Intel MKL库链接

- 动态链接myprog.f:

```
ifort myprog.f –lmkl_rt
```

使用Fortran 95 LAPACK接口和LP64接口的并行Intel MKL库链接

- 静态链接myprog.f:

```
ifort myprog.f -L$MKLPATH -I$MKLINCLUDE \
-I$MKLINCLUDE/intel64/lp64 -lmkl_lapack95_lp64 \
-Wl,--start-group $MKLPATH/libmkl_intel_lp64.a \
$MKLPATH/libmkl_intel_thread.a $MKLPATH/libmkl_core.a \
-Wl,--end-group -liomp5 -lpthread -lm
```

使用Fortran 95 BLAS接口和LP64接口的并行Intel MKL库链接

- 静态链接myprog.f:

```
ifort myprog.f -L$MKLPATH -I$MKLINCLUDE \
-I$MKLINCLUDE/intel64/lp64 -lmkl_blas95_lp64 \
-Wl,--start-group $MKLPATH/libmkl_intel_lp64.a \
$MKLPATH/libmkl_intel_thread.a $MKLPATH/libmkl_core.a \
-Wl,--end-group -liomp5 -lpthread -lm
```

在命令行上列出所需库链接

```
<files to link>

-L<MKL path> -I<MKL include>
[-I<MKL include>/{ia32|intel64|{ilp64|lp64}}]
[-lmkl_blas{95|95_ilp64|95_lp64}]
[-lmkl_lapack{95|95_ilp64|95_lp64}]
[ <cluster components> ]
-lmkl_{intel|intel_ilp64|intel_lp64|intel_sp2dp|gf|gf_ilp64|gf_lp64}
-lmkl_{intel_thread|gnu_thread|pgi_thread|sequential}
-lmkl_core
-liomp5 [-lpthread] [-lm] [-ldl]
```

- *[]*内的表示可选，*|*表示其中之一、*{}*表示含有。
- 在静态连接时，在分组符号（如，*-Wl,--start-group␣$MKLPATH/libmkl_cdft_core.a␣$MKLPATH/libmkl_blacs_intelmpi_ilp64 .a␣$MKLPATH/libmkl_intel_ilp64.a␣$MKLPATH/libmkl_intel_thread.a $MKLPATH/libmkl_core.a␣-Wl,--end-group*）封装集群组件、接口、线程和计算库。

动态选择接口和线程层链接

SDL接口使得用户可以动态选择Intel MKL的接口和线程层。

设置接口层

- 可用的接口与系统架构有关，对于Intel 64架构，可使用LP64和ILP64接口
- 在运行时设置接口，可调用mkl_set_interface_layer函数或设置*MKL_INTERFACE_LAYER*环境变量
- 可用的接口层的值

接口层	*MKL_INTERFACE_LAYER*的值	mkl_set_interface_layer的参数值
LP64	LP64	MKL_INTERFACE_LP64
ILP64	ILP64	MKL_INTERFACE_ILP64

- 如果调用了mkl_set_interface_layer函数，那么环境变量*MKL_INTERFACE_LAYER*的值将被忽略
- 默认使用LP64接口

设置线程层

- 在运行时设置线程层，可以调用mkl_set_threading_layer函数或者设置环境变量*MKL_THREADING_LAYER*
- 可用的线程层的值

线程层	*MKL_INTERFACE_LAYER*的值	mkl_set_interface_layer的参数值
Intel线程	INTEL	MKL_THREADING_INTEL
串行线程	SEQUENTIAL	MKL_THREADING_SEQUENTIAL
GNU线程	GNU	MKL_THREADING_GNU
PGI线程	PGI	MKL_THREADING_PGI

- 如果调用了mkl_set_threading_layer函数，那么环境变量MKL_THREADING_LAYER的值被忽略
- 默认使用Intel线程

使用ILP64接口 vs. LP64接口

- Intel MKL ILP64库采用64-bit整数（索引超过含有$2^{31}-1$个元素的大数组时使用），而LP64库采用32-bit整数索引数组
- LP64和ILP64接口在接口层实现，分别采用下面接口层链接：
  - 静态链接：libmkl_intel_lp64.a或libmkl_intel_ilp64.a
  - 动态链接：libmkl_intel_lp64.so或libmkl_intel_ilp64.so
- ILP64接口提供以下功能：
  - 支持大数据数组（具有超过$2^{31}-1$个元素）
  - 添加-i8编译器参数编译Fortran程序
- LP64接口提供与以前Intel MKL版本的兼容，因为LP64对于仅提供一种接口的版本低于9.1的Intel MKL来说是一个新名字
- 如果用户的应用采用Intel MKL计算大数据数组或此库也许在将来会用到时请选择使用ILP64接口
- Intel MKL提供的ILP64和LP64头文件路径是相同的

采用LP64/ILP64编译

- 采用ILP64和LP64接口进行编译：
  - Fortran：
    - ILP64： *ifort 」–i8」–I<mkl」directory>/include」 ...*
    - LP64： *ifort 」–I<mkl」directory>/include」 ...*
  - C/C++：
    - ILP64： *icc 」–DMKL_ILP64」–I<mkl」directory>/include」...*
    - LP64： *icc 」–I<mkl」directory>/include」 ...*
- 注意：采用-i8或-DMKL_ILP64选项链接LP64接口库时也许将会产生预想不到的错误。

编写代码

- 如果不使用ILP64接口，无需修改代码。
- 为了移植或者新写代码使用ILP64接口，需要使用正确的Intel MKL函数和子程序的参数类型：

整数类型	Fortran	C/C++
32-bit整数	INTEGER*4或INTEGER(KIND=4)	int
针对ILP64/ LP64的通用整数（ILP64使用64-bit，其余32-bit）	INTEGER，不指明KIND	MKL_INT
针对ILP64/ LP64的通用整数（64-bit整数）	INTEGER*8或INTEGER(KIND=8)	MKL_INT64
针对ILP64/LP64的FFT接口	INTEGER，不指明KIND	MKL_LONG

局限性

所有Intel MKL函数都支持ILP64编程，但是针对Intel MKL的FFTW接口：

- FFTW 2.x封装不支持ILP64
- FFTW 3.2封装通过专用功能函数plan_guru64支持ILP64

使用Fortran 95接口库

- libmkl_blas95*.a和libmkl_lapack95*.a库：
  - 分别含有BLAS和LAPACK所需的Fortran 95接口
  - 并且是与编译器无关
- 在Intel MKL包中：
  - 已经为Intel Fortran编译器预编译了
  - 如果使用其它编译器，请在使用前先编译

使用线程库链接 I

- 串行库模式
  - 采用Intel MKL串行（非线程化）模式时，Intel MKL运行非线程化代码。它是线程安全的（除了LAPACK已过时的子程序?lacon），即可以在用户程序的OpenMP代码部分使用。串行模式不要求与OpenMP运行时库的兼容，环境变量*OMP_NUM_THREADS*或其Intel MKL等价变量对其也无影响。
  - 只有在不需要使用Intel MKL线程时才应使用串行模式。当使用一些非Intel编译器线程化程序或在需要非线程化库（比如使用MPI的一些情况时）的情形使用Intel MKL时，串行模式也许有用。为了使用串行模式，请选择*sequential.*库。
  - 对于串行模式，由于*sequential.*依赖于pthread，请在链接行添加POSIX线程库(pthread)。
- 选择线程库层

  一些Intel MKL支持的编译器使用OpenMP线程技术。Intel MKL支持这些编译器提供OpenMP技术实现，为了使用这些支持，需要采用正确的线程层和编译器支持运行库进行链接。

使用线程库链接 II

- 线程层：每个Intel MKL线程库包含针对同样的代码采用不同编译器（Intel、GNU和PGI编译器）分别编译的库
- 运行时库：
  - 此层包含Intel编译器兼容的OpenMP运行时库libiomp
  - 在Intel编译器之外，libiomp提供在Linux操作系统上对更多线程编译器的支持
  - 采用GNU编译器线程化的程序可以安全地采用intel MKL和libiomp链接
- 不同情形使用Intel MKL时选择线程库和运行时库（仅静态链接情形）

编译器	应用是否线程化	线程层	推荐的运行时库	备注
Intel	无所谓	libmkl_intel_thread.a	libiomp5.so	
PGI	Yes	libmkl_pgi_thread.a 或libmkl_sequential.a	由PGI*提供	使 用libmkl_sequential.a从Intel MKL调用中去除线程化
PGI	No	libmkl_intel_thread.a	libiomp5.so	
PGI	No	libmkl_pgi_thread.a	由PGI*提供	
PGI	No	libmkl_sequential.a	None	
GNU	Yes	libmkl_gnu_thread.a	libiomp5.so或GNU OpenMP运行时库	libiomp5提供监控缩放性能
GNU	Yes	libmkl_sequential.a	None	
GNU	No	libmkl_intel_thread.a	libiomp5.so	
other	Yes	libmkl_sequential.a	None	
other	No	libmkl_intel_thread.a	libiomp5.so	

使用计算库链接

- 如不使用Intel MKL集群软件在链接应用程序时只需要一个计算库即可，其依赖于链接方式：
  - 静态链接：libmkl_core.a
  - 动态链接：libmkl_core.so
- 采用Intel MKL集群软件的计算库
  - ScaLAPACK和集群Fourier变换函数(Cluster FFTs)要求更多的计算库，其也许依赖于架构
  - 针对Intel 64架构的使用ScaLAPACK或集群FFTs的计算库：

函数域	静态链接	动态链接
ScaLAPACK，LP64接口	libmkl_scalapack_lp64.a和 libmkl_core.a	libmkl_scalapack_lp64.so和 libmkl_core.so
ScaLAPACK，ILP64接口	libmkl_scalapack_ilp64.a和 libmkl_core.a	libmkl_scalapack_ilp64.so和 libmkl_core.so
集群FFTs	libmkl_cdft_core.a和 libmkl_core.a	libmkl_cdft_core.so和 libmkl_core.so

- 针对IA-32架构的使用 ScaLAPACK或集群FFTs的计算库：

函数域	静态链接	动态链接
ScaLAPACK	libmkl_scalapack_core.a和libmkl_core.a	libmkl_scalapack_core.so和libmkl_core.so
集群FFTs	libmkl_cdft_core.a和libmkl_core.a	libmkl_cdft_core.so和libmkl_core.so

- 对于ScaLAPACK和集群FFTs，当在MPI程序中使用时，还需要添加BLACS库

使用编译器运行库链接

- 在静态链接其它库时，也可动态链接libiomp5、兼容的OpenMP运行时库
- 静态链接libiomp5也许会存在问题：
  - 因为由于操作环境或应用越复杂，将会包含更多多余的库的复本
  - 不仅会导致性能问题，甚至导致不正确的结果
- 动态链接libiomp5时，需确保*LD_LIBRARY_PATH*环境变量设置正确

使用系统库链接

- 使用Intel MKL的FFT、Trigonometric Transforn或Poisson、Laplace和Helmholtz求解程序时，需要通过在链接行添加-lm参数链接数学支持系统库
- 在Linux系统上，由于多线程libiomp5库依赖于原生的pthread库，因此，在任何时候，libiomp5要求在链接行随后添加-lpthread参数（列出的库的顺序非常重要）

1 Intel C/C++、Fortran编译器简介
- Intel编译器安装配置
- Intel C/C++编译器用法
- Intel Fortran编译器用法
- Intel调试器用法

2 GNU C/C++、Fortran编译器简介
- GNU C/C++编译器用法
- GNU Fortran编译器用法
- GNU程序调试器用法

3 数学函数库
- Intel MKL数学函数库
- 其余数学统计函数库

4 联系信息

ACML： AMD核心数学库

AMD Core Math Library(ACML)是一些数值函数的组合，并且特别针对AMD64平台处理器（如Opteron）等做了优化。ACML函数库具有FORTRAN 77和C语言接口，主要包含：

- BLAS - 基本线性代数子系统库
- LAPACK - 线性代数库
- FFT - 傅立叶变换程序
- RNG - 随机数发生器和统计分布函数

网址：http://developer.amd.com/tools-and-sdks/cpu-development/amd-core-math-library-acml/

Rogue Wave IMSL：数学与统计数值函数库

- IMSL是Rogue Wave公司的一套完整的数学与统计数值函数库，能够让使用者嵌入至他们的应用系统中
- IMSL提供高效能的计算能力并提供所有专家所需要开发与建置的精密数学分析应用程序。这些程序库能够让用户直接使用已经写好的数学与统计算法，而免去自我撰写程序的麻烦，并能够让用户轻易的嵌入至用户所使用的C与Fortran语言程序中
- IMSL与MKL相比主要多了积分、随机数及统计等方面的函数
- 网址：www.roguewave.com/products/imsl-numerical-libraries.aspx

Netlib: 数学软件、论文及数据库的宝库

- 免费开源
- http://www.netlib.org

Netlib Libraries:

a
access
aicm
alliant
amos
ampl
anl-reports
apollo
arpack
atlas
benchmark
bib
bibnet
bihar
blacs
blas
blast
bmp
c
c++
fftpack
fishpack
fitpack
floppy
fmm
fn
fortran
fortran-m
fp
gcv
gmat
gnu
go
graphics
harwell
hence
hompack
hpf
hypercube
ieeecss
machines
magic
maspar
mds
microscope
minpack
misc
mpfun
mpi
mpicl
na-digest-html
napack
netsolve
news
numeralgo
ode
odepack
odrpack
opt
p4
research
rib
scalapack
sched
scilib
seispack
sequent
sfmm
slap
slatec
sminpack
sodepack
sparse
sparse-blas
sparspak
specfun
spin
srwn
stoeplitz
stringsearch
cephes
chammp
cheney-kincaid
clapack
commercial
confdb
conformal
contin
control
crc
cumulvs
ddsv
dierckx
diffpack
domino
eispack
elefunt
env
etemplates
f2c
fdlibm
ijsa
image
intercom
itpack
jakef
java
kincaid-cheney
la-net
lanczos
lanz
lapack
lapack++
lapack3e
lapack90
laso
lawson-hanson
linalg
linpack
list
lp
lyapack
paragraph
paranoia
parkbench
parmacs
pascal
pdes
performance
photo
picl
pltmg
poly2
polyhedra
popi
port
posix
pppack
presto
problem-set
pvm3
quadpack
random
svdpack
templates
tennessee
textbook
toeplitz
toms
tomspdf
transform
typesetting
uncon
vanhuffel
vfftpack
vfnlib
voronoi
xblas
xmagic
xmetlib
y12m

GotoBLAS2、Open BLAS及ATLAS

- GotoBLAS2性能非常高的一种BLAS实现，Open BLAS是其后续版本
  - GotoBLAS2：https://www.tacc.utexas.edu/tacc-projects/gotoblas2/
  - Open BLAS：http://xianyi.github.io/OpenBLAS/
- ATLAS(Automatically Tuned Linear Algebra Software)：
  - 性能非常高另一种BLAS实现
  - 网址：http://math-atlas.sourceforge.net/

联系信息

- 中国科大超级计算中心：
  - 电话：0551-63602248
  - 信箱：sccadmin@ustc.edu.cn
  - 主页：http://scc.ustc.edu.cn
  - 办公室：中国科大东区新图书馆一楼东侧126室
- 李会民：
  - 电话：0551-63600316
  - 信箱：hmli@ustc.edu.cn
  - 主页：http://hmli.ustc.edu.cn