How do I compile with MKL Libraries on HiPAS?

Order Matters When Compiling and (especially) Linking

The order of arguments, flags, libraries, and files matters when compiling and linking; we can must be careful to link the MKL libraries in the correct order.

Putting the include directories (e.g. "-I /opt/intel/include") in the correct place in the command line may matter as well.

Here are a few pages about why that matters:

  - Most detailed, but best explanations
  - Fairly detailed, but more concise than the above.
  - Quick-n-dirty example and explanation
  - Quote: "Most current linkers will search all libraries, regardless of order, but since some do not do this..." --> Does our linker need to be updated?

So, the Makefile needed some adjusting to get all its flags in the correct order.

MKL Libraries: Care Needed

There are a few dozen separate MKL libraries. There are a few main libraries times several versions per library (e.g. for different processors, different thread models, etc.). Care is needed to know which specific libraries are needed for a given project. Some sources claim that it's as easy as adding "-mkl" to the compile command...but this doesn't appear to hold true in practice. Fortunately, Intel offers an online [MKL Link Line Assistant](http://www.intel.com/mkl)
tool to figure out what the linking line should look like. It really does help — I would encourage anyone using the MKL libraries to use this tool early in their work. It helps in two ways: first, it shows the correct library order (see above); second, it helps one know which versions of the libraries are needed (e.g. libmkl_lapack95.a vs. libmkl_lapack95_lp64.a vs. libmkl_lapack95_ilp64.a).

**Version of MKL and Setting Environment Variables**

Anyone using any of Intel's compilers (icc, ifort) and/or MKL should make sure to run the compilervars.sh (or equivalent) script. Intel has an article about it [here](#). Here's what they need to know about it:

- Each installed version of Intel's software (e.g. 11.0 vs. 11.1 on the cluster) has its own compilervars.sh script. Either should work fine. Usually, one should use the most up-to-date version.
- When running it, one must include the processor argument (intel64 for the cluster and most everyone's workstations). The following should work on the cluster in descending order of specificity, though they should all have the same effect:
  - `source /opt/intel/bin/compilervars.sh -arch intel64 -platform linux`
  - `source /opt/intel/bin/compilervars.sh -arch intel64`
  - `source /opt/intel/bin/compilervars.sh intel64`
- That script should be run prior to any compilation/linking of code. (!!!!)
- It only needs to be run once per session to set environment variables correctly — not every time ifort is called.
- Running that script in one terminal affects only that terminal session. If multiple terminal windows are open, a user might want to run it in each terminal.
  - Better yet, add it to one's .bashrc or .cshrc file and have it run automatically for each session.