Lab 1 - Instructions and Summary

Compiling a MPI program

Load Intel compilers

```
module load intel/psxe-2015
```

Compile for speed `-Ofast` and add information about line numbers `-g`

```
mpicc -g -Ofast -o myprogram myprogram.c
mpifc -g -Ofast -o myprogram myprogram.f90
```

Compile for debugging `-O0` and add information about line numbers `-g`

```
mpicc -g -O0 -o myprogram myprogram.c
mpifc -g -O0 -o myprogram myprogram.f90
```

Running an MPI program on Yeti

Job script for an MPI run on normal partion

```
#!/bin/bash
#SBATCH -p normal
#SBATCH -t 0:02:00
#SBATCH -N 2
#SBATCH --job-name=mpi_hello
#SBATCH --output=out.mpi_hello
#SBATCH --ntasks-per-node=20

module load intel/psxe-2015

srun --mpi=pmi2 ./mpi_hello
```

Writing a parallel "hello world" program

- All process print: "Hello World from process = %d processor %s"
- Processor 0 prints: "Number of mpi processes = %d"

Fortran 90

```
use mpi
MPI_COMM_WORLD
MPI_INIT(IERROR) ! integer argument
MPI_COMM_SIZE(COMM, SIZE, IERROR) ! all arguments are integer
MPI_COMM_RANK(COMM, RANK, IERROR) ! all arguments are integer
MPI_FINALIZE(IERROR) ! integer argument
```

C

```
MPI_Comm MPI_COMM_WORLD
int MPI_Init(int *argc, char ***argv)
int MPI_Comm_size(MPI_Comm comm, int *size)
int MPI_Comm_rank(MPI_Comm comm, int *rank)
int MPI_Finalize()
```

Writing a MPI version of the PI program

- Process 0 reads the number of integration intervalls
- Process 0 broadcasts this number to the rest of the processes
- Each process computes its share of PI
- Process 0 receives all shares and calculates an approximation of pi
- Process 0 prints the result and the error

Fortran 90

```
MPI_BCAST(BUFFER, COUNT, DATATYPE, ROOT, COMM, IERROR)
    <type>    BUFFER(*)
    INTEGER    COUNT, DATATYPE, ROOT, COMM, IERROR
MPI_INTEGER, MPI_DOUBLE_PRECISION
MPI_REDUCE(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, ROOT, COMM,
        IERROR)
    <type>    SENDBUF(*), RECVBUF(*)
    INTEGER    COUNT, DATATYPE, OP, ROOT, COMM, IERROR
```

C

```
int MPI_Bcast(void *buffer, int count, MPI_Datatype datatype,
    int root, MPI_Comm comm)
MPI_INT, MPI_DOUBLE
int MPI_Reduce(void *sendbuf, void *recvbuf, int count,
    MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)
```