PRACE AUTUMN SCHOOL 2016
September 27-30, 2016
Hagenberg, Austria

PRACE Autumn School 2016
PETSc tutorial

Part II: Hello World

Václav Hapla
IT4Innovations
VSB– Technical University of Ostrava
Ostrava, Czech Republic

What headers to include?

- You can include **all PETSc headers** at once by
```
#include "petsc.h" //includes all PETSc headers
```
- Or you can include **specific headers**
```
#include "petscsys.h" //framework routines
#include "petscvec.h" //vectors
#include "petscmat.h" //matrices
```
- **Higher** level headers **include** all **lower** level headers needed
```
#include "petscksp.h" //includes vec,mat,dm,pc
```

Initialize & Finalize (1)

```
static char help[] = "Empty program.\n\n";
#include <petscsys.h>
int main(int argc,char **argv)
{
  PetscErrorCode ierr;
  ierr = PetscInitialize(&argc,&argv,(char *)0,help);CHKERRQ(ierr);
  ierr = PetscFinalize();CHKERRQ(ierr);
  return 0;
}
```

- Every PETSc program begins with the call to `PetscInitialize()`
- ends with the call to `PetscFinalize()`
- they call `MPI_Init()`, `MPI_Finalize()`

Initialize & Finalize (2)

```
static char help[] = "Empty program.\n\n";
#include <petscsys.h>
int main(int argc,char **argv)
{
  PetscErrorCode ierr;
  ierr = PetscInitialize(&argc,&argv,(char *)0,help);CHKERRQ(ierr);
  ierr = PetscFinalize();CHKERRQ(ierr);
  return 0;
}
```

- `argc,argv` - pointer to **command line arg count** and **array**, respectively; PETSc filters out and parses PETSc options
- `help` - additional **help messages** to print when the executable is invoked with the command line arg `"-help"`
- handling of PETSc options will be discussed later

Communicators

- **communicator** (in MPI) = an opaque object of `MPI_Comm` type that defines **process group** and **synchronization channel**
- PETSc built-in communicators:
  - `PETSC_COMM_SELF` ⇒ just this process ⇒ for serial objects
  - `PETSC_COMM_WORLD` ⇒ all processes ⇒ for parallel objects
- you can define your own communicators
- MPI can split communicators, spawn processes on new communicators – PETSc does not deal with that

Error handling

- PETSc is written in C
- C has **no support** for **exceptions** (it's a C++ feature)
- instead of throwing exception, every routine returns **integer error code** (`PetscErrorCode` type)
- similarly to MPI
- error code is **„catched“ by** `CHKERRQ` **macro**

```
PetscErrorCode ierr;
ierr = SomePetscRoutine();CHKERRQ(ierr);
```

Utility routines (1)

- PETSc provides many useful utilities
- prefixed by `Petsc`
- **parallel flow control:**
  `Barrier, SequentialPhaseBegin/End`
- **memory management and checking:**
  `Malloc, Free, MallocValidate, MallocDump`

Utility routines (2)

- **logging:**
  `PetscLogEventRegister/Begin/End`
- **string handling:**
  `Strcat/cmp/cpy/len/tolower/replace/ToArray`
- **MATLAB engine interface:**
  `MatlabEngineCreate/Destroy/Evaluate`
- and **many more**

Primitive datatypes

- PETSc provides its own **primitive data types**

```
PetscInt n = 20;
PetscScalar v = -3.5, w = 3.1e9;
PetscReal x = 2.55, y = 1e-9;
```

- It is better to use them instead of **built-in C types**

$\Rightarrow$ better **portability**

$\Rightarrow$ easy switching between **real** and **complex scalars**

$\Rightarrow$ easy switching between **32-bit** and **64-bit** numbers

Options (1)

- PETSc provides routines for managing the **options database**
- `-help` option prints help to all built-in options relevant for a given program
- in your program, you can call routines
  `PetscOptionsGetInt`
  `PetscOptionsGetString`
  `PetscOptionsGetReal`
  `...`
  to obtain the values

Options (2)

Example:

- in **command-line**

```
./myapp -myint 10 -myreal 1e3
```

- in **program** myapp:

```
PetscReal myreal;
PetscInt  myint;
PetscOptionsGetInt( NULL,"-myint",  &myint, NULL);
PetscOptionsGetReal(NULL,"-myreal", &myreal,NULL);
// now myint=10, myreal=1e3
```

Ways to set options

- command line
- filename in the third argument of `PetscInitialize()`
- `~/.petscrc`
- `$PWD/.petscrc`
- `$PWD/petscrc`
- `PetscOptionsInsertFile()`
- `PetscOptionsInsertString()`
- `PETSC_OPTIONS` environment variable
- command line option `-options_file [file]`

Print to standard output

`PetscPrintf`

- prints to standard output only from the **zeroth rank** in the communicator comm

C

```
PetscErrorCode
  PetscPrintf(MPI_Comm, const char format[], ...)
```

F

```
PetscPrintf(MPI_Comm, character(*), PetscErrorCode)
```

  - only single string can be passed

Synchronized print

To obtain output from **all processors** in **one-after-the-other way**, one can call:

```
PetscSynchronizedPrintf(PETSC_COMM_WORLD,
      "Hello World from %d\n",rank);
PetscSynchronizedFlush(PETSC_COMM_WORLD,PETSC_STDOUT);
```

Output:

```
Hello World from 0
Hello World from 1
Hello World from 2
```

PETSc Hello world in F

```
program main
    integer ierr, rank
#include "include/finclude/petsc.h"
    call PetscInitialize(PETSC_NULL_CHARACTER, ierr)
    call MPI_Comm_rank(PETSC_COMM_WORLD, rank, ierr)
    if (rank .eq. 0) then
        print *, 'Hello World from ', rank
    endif
    call PetscFinalize(ierr)
    end
```

PETSc Hello world in C

```
static char help[] = "Hello world program.\n\n";
#include <petscsys.h>
int main(int argc,char **argv)
{
  PetscErrorCode ierr;
  PetscMPIInt    rank;
  PetscInitialize(&argc,&argv,(char *)0,help);
  MPI_Comm_rank(PETSC_COMM_WORLD,&rank);
  PetscPrintf(PETSC_COMM_SELF,"Hello World from %d\n",rank);
  PetscFinalize();
  return 0;
}
```

PETSc Hello world in C - with error checking

```
static char help[] = "Hello world program.\n\n";
#include <petscsys.h>
int main(int argc,char **argv)
{
  PetscErrorCode ierr;
  PetscMPIInt    rank;
  ierr = PetscInitialize(&argc,&argv,(char *)0,help);CHKERRQ(ierr);
  ierr = MPI_Comm_rank(PETSC_COMM_WORLD,&rank);CHKERRQ(ierr);
  ierr = PetscPrintf(PETSC_COMM_SELF,"Hello World from %d\n",rank);CHKERRQ(ierr);
  ierr = PetscFinalize();
  return 0;
}
```

Hands-on: setup

- log to Anselm: `ssh anselm.it4i.cz`
- submit your interactive job:
  `qsub -I -A DD-16-44 -q R1303343`
- `module avail petsc`
- `module avail slepc`
- `module load slepc/3.7.2-icc16-impi5-mkl-dbg`
- `echo $PETSC_DIR; echo $PETSC_ARCH; echo $SLEPC_DIR`

Hands-on: setup

- log to Anselm: `ssh anselm.it4i.cz`
- submit your interactive job:
  `qsub -I -A DD-16-44 -q R1303343`
- `module avail petsc`
- `module avail slepc`
- `module load slepc/3.7.2-icc16-impi5-mkl-dbg`
- `echo $PETSC_DIR; echo $PETSC_ARCH; echo $SLEPC_DIR`
- `cp -uRT /apps/libs/petsc/pas16 ~/petsc`
- `cd ~/petsc/exercises`

Hands-on: ex1.c

1. `cp -RT /apps/libs/petsc/pas16 ~/petsc`
2. `cd ~/petsc/exercises`
3. `make ex1`
4. `mpirun -n 4 ./ex1`
5. notice direct calls to MPI are possible
6. compare the error output for `PetscPrintf` with and without CHKERRQ
7. fix the error (change the communicator to `PETSC_COMM_WORLD`)
8. add similar call to `PetscPrintf`, but now with PETSC_COMM_SELF communicator – what's the problem?
9. add similar call to `PetscSynchronizedPrintf/PetscSynchronizedFLush`
10. set the value of variable `myint` from command line option `-myint` and print its value
11. try setting the same from the `petscrc` file and overriding the value from command line

Thanks for your attention.