SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ZAVOD ZA ELEKTRONIKU, MIKROELEKTRONIKU,
RAČUNALNE I INTELIGENTNE SUSTAVE

DIPLOMSKI STUDIJ
PARALELNO PROGRAMIRANJE

Upute za izradu domaćih zadaća - MPI

Domagoj Jakobović, Karlo Knežević

Verzija: *1.2, 8.3.2023.*

Zagreb, ožujak 2023.

SADRŽAJ

1. MPI

MPI (engl. *Message-Passing Interface*) jest sučelje specifikacije biblioteke za mehanizam razmjene poruka. MPI ponajviše predstavlja paralelni programski model razmjene poruka, gdje se podaci prebacuju iz adresnog prostora jednog procesa u adresni prostor drugoga, kroz surađujuće operacije na svakom procesu.

MPI je specifikacija, a ne implementacija. Postoji više MPI implementacija s obzirom na programski jezik. Sam MPI standard je specifikacija sučelja biblioteke koje će se koristiti kao funkcije, podrutine ili metode s obzirom na implementaciju MPI-a u nekom programskom jeziku. C, C++, Fortran-77 i Fortran-95 su dio MPI standarda. Cilj MPI-ja je razvoj široko upotrebljivog standarda za pisanje programa s principom razmjena poruka. Trenutna aktivna specifikacija standarda MPI je 4.0.

Neke od implementacija MPI specifikacije su MPICH, OpenMPI te MS-MPI.

1.1. Instalacija

Kako bi se mogao napisati, prevesti i izvršiti paralelan program koji koristi MPI funkcije potrebni su: biblioteka zaglavlja MPI funkcija, jezični prevoditelj, biblioteka implementiranih MPI funkcija te pokretač MPI programa. Kako bi se preveo MPI program, može se koristiti jezični prevoditelj namijenjen jeziku u kojem je program pisan ili program omotač (engl. *wrapper*) koji ima unaprijed postavljene zastavice za MPI biblioteke prilikom prevođenja. Nakon što se MPI program prevede, izvršni program pokreće se iz komandne ljuske naredbom *mpiexec*. Naredba *mpiexec* stvara i pokreće onoliko MPI procesa koliko je zadano u argumentima.

U nastavku slijedi opis instalacije MPI biblioteka za pojedine operacijske sustave.

1.1.1. Windows

Preporučena implementacija MPI specifikacije za Windows operacijski sustav jest MS-MPI. Sa stranice za preuzimanje preuzmite sljedeće instalacijske pakete: msmpisdk.msi

i msmpisetup.exe. Prije preuzimanja provjerite zadovoljava li računalo na koje namjeravate instalirati navedene programe ograničenja. Nakon preuzimanja oba instalacijska paketa, pokrenite oba paketa u neovisnom redoslijedu i pratite korake instalacije.

Nakon što su uspješno instalirana oba paketa, na računalu se nalaze dva direktorija: C:\Program Files\Microsoft MPI i C:\Program Files (x86)\Microsoft SDKs\MPI. Na prvoj putanji nalazi se pokretač MPI procesa (mpiexec), a na drugoj putanji nalazi se zaglavne biblioteke MPI funkcija i izvršne biblioteke MPI funkcija (lib i bin). Prilikom instalacije, putanje do pokretača MPI procesa, zaglavlja MPI funkcija te izvršnih biblioteka MPI funkcija za 32-bitnu i 64-bitnu arhitekturu dodane su u varijable okruženja.

Kako biste provjerili uspješnost instalacije MPI okoline, u naredbenu ljusku unesite sljedeću naredbu:

```
C:\Users\PP\Desktop> mpiexec
```

Ako je instalacija MPI okoline i postavljanje varijabli okruženja uspješno, tada biste kao odgovor trebali vidjeti sljedeći ispis:

```
Microsoft MPI Startup Program [Version 9.0.12497.9]

Launches an application on multiple hosts.

Usage:

mpiexec [options] executable [args]
[ : [options] exe [args] : ... ]
mpiexec -configfile <file name>

Common options:

-n <num_processes>
-env <env_var_name> <env_var_value>
-wdir <working_directory>
-hosts n host1 [m1] host2 [m2] ... hostn [mn]
-cores <num_cores_per_host>
-lines
-debug [0-3]
```

```
Examples:

mpiexec -n 4 pi.exe
mpiexec -hosts 1 server1 master : -n 8 worker

For a complete list of options, run mpiexec -help2
For a list of environment variables, run mpiexec -help3

You can reach the Microsoft MPI team via
email at askmpi@microsoft.com
```

U nastavku slijedi opis postavljanja integrirane razvojne okoline u kojoj ćete pisati MPI program. Pretpostavlja se da koristite C programski jezik i razvojnu okolinu Visual Studio. Unutar Visual Studija napravite prazan projekt za programski jezik C. Stvorite *main.c* datoteku i u nju kopirajte primjer Pozdrav svijete 1.2.

Primjetit ćete da su sve MPI funkcije, uključujući i MPI zaglavlje podcrtani. Uzrok ovih grešaka jest činjenica da razvojna okolina ne prepoznaje putanju do MPI zaglavlja, a time i do prototipova MPI funkcija korištenih u kodu. Kako bi ispravili ovu grešku, prevoditelju je potrebno zadati gdje se nalaze zaglavlja MPI funkcija (mpi.h).

Unutar programskog paketa (engl. *Visual Studio solution*), desnim klikom miša na projekt otvorite postavke projekta. Na početku prozora nalazi se konfiguracijsko polje projekta (engl. *configuration*). Preporuka je odabrati sve konfiguracije projekta, konfiguraciju za otklanjanje pogrešaka (engl. *debug*) i distribucijsku konfiguraciju (engl. *release*), (engl. *all configurations*). Ako odaberete samo jednu konfiguraciju, tada prilikom prevođenja i stvaranja izvršnog MPI programa koristite tu konfiguraciju.

U postavkama za C/C++, u polje dodatnih direktorija za uključivanje zaglavnih datoteka (engl. *additional include directories*) dodajte putanju do direktorija u kojem se nalazi *mpi.h* datoteka, primjerice C:\Program Files (x86)\Microsoft SDKs\MPI\Include. Nakon što ste dodali ovu putanju i potvrdili izmjene u postavkama (engl. *apply*), razvojna okolina ne bi smjela više pokazivati pogreške u programu. U ovom trenutku, zadovoljene su pretpostavke za uspješno prevođenje programa u objektnu datoteku.

Ako pokrenete prevođenje, primjetit ćete da i dalje postoje greške. U ovom trenutku greške javlja program povezivač (engl. *linker*) koji ne može povezati objekte datoteke prevedenog programa i izvršne biblioteke MPI funkcija u izvršnu datoteku. Razlog ovakvih grešaka jest činjenica da program povezivač ne zna gdje pronaći izvršne datoteke MPI funkcija. Vratite se ponovno u postavke projekta. U postavkama za

program povezivač, u polje dodatnih direktorija biblioteka (engl. *additional library directories*) dodajte putanju do izvršnih biblioteka MPI funkcija, primjerice C:\Program Files (x86)\Microsoft SDKs\MPI\Lib. Primjetit ćete da se unutar Lib direktorija nalaze direktoriji x86 i x64. Navedeni direktoriji sadrže prevedene implementacije MPI specifikacije za 32-bitnu i 64-bitnu arhitekturu. Odaberite postavke ulaza u program povezivač (engl. *input*) i u polje dodatnih međuovisnosti (engl. *additional dependencies*) odaberite relativnu putanju do one izvršne biblioteke koja je prilagođena arhitekturi za koju ćete prevesti program. Ako prevodite za 32-bitnu arhitekturu, tada je putanja x86/msmpi.lib, a u suprotnom je x64/msmpi.lib.

Nakon što ste izmijenili postavke za program povezivač, možete uspješno stvoriti prvi MPI izvršni program.
Nakon što stvorite izvršni MPI program, pokrenite 5 MPI procesa koji će ispisivati poruku pozdrava svijetu:

```
C:\Users\PP\Desktop> mpiexec.exe -n 5 .\HelloWorld.exe
```

U tom slučaju pojavio se sljedeći ispis:

```
Hello world from processor PP, rank 2 out of 5 processors
Hello world from processor PP, rank 3 out of 5 processors
Hello world from processor PP, rank 1 out of 5 processors
Hello world from processor PP, rank 4 out of 5 processors
Hello world from processor PP, rank 0 out of 5 processors
```

Ako na operacijskom sustavu Windows koristite drugu razvojnu okolinu, tada u razvojnoj okolini koju koristite morate podeciti putanje do zaglavne MPI datoteke (mpi.h) i do izvršne biblioteke MPI specfikacije (msmpi.lib).

Ako ste instalirali neku drugu implementaciju MPI specifikacije, tada provjerite jesu li ispravno postavljene varijable okruženja i ispravno postavite prethodno navedene putanje u razvojnoj okolini koju koristite.

Ako na operacijskom sustavu Windows koristite neki drugi programski jezik (Python, Java, C#), tada morate preuzeti MPI biblioteke prilagođene za navedene programske jezike.

1.1.2. Linux i OS X

Za potrebe rada na ovim operacijskim sustavima, preporuča se koristiti implementacije MPICH ili OpenMPI. Na stranicama dotičnih implementacija dostupne su upute

za postavljanje radne okoline, ovisno o odabranom programskom jeziku i razvojnoj okolini.

1.2. Primjeri

Hello world

```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
  // Initialize the MPI environment
  MPI_Init(NULL, NULL);

  // Get the number of processes
  int world_size;
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);

  // Get the rank of the process
  int world_rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

  // Get the name of the processor
  char processor_name[MPI_MAX_PROCESSOR_NAME];
  int name_len;
  MPI_Get_processor_name(processor_name, &name_len);

  // Print off a hello world message
  printf("Hello world from processor %s, rank %d"
    " out of %d processors\n",
    processor_name, world_rank, world_size);

  // Finalize the MPI environment.
  MPI_Finalize();
}
```

Slanje i primanje poruka

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
```

```
  // Initialize the MPI environment
  MPI_Init(NULL, NULL);
  // Find out rank, size
  int world_rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
  int world_size;
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);

  // We are assuming at least 2 processes for this task
  if (world_size < 2) {
    fprintf(stderr, "World␣size␣must␣be␣greater␣than␣1␣for␣%s\n",
        argv[0]);
    MPI_Abort(MPI_COMM_WORLD, 1);
  }

  int number;
  if (world_rank == 0) {
    // If we are rank 0, set the number to -1 and send it to
       process 1
    number = -1;
    MPI_Send(&number, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
  }
  else if (world_rank == 1) {
    MPI_Recv(&number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
       MPI_STATUS_IGNORE);
    printf("Process␣1␣received␣number␣%d␣from␣process␣0\n",
       number);
  }
  MPI_Finalize();
}
```