Osnovne MPI I/O funkcije

• Osnovne MPI I/O funkcije liče na standardne funkcije jezika C:

**• int MPI\_File\_open (MPI\_Comm comm, const char \*filename, int amode, MPI\_Info info, MPI\_File \*fh)**

**• int MPI\_File\_seek (MPI\_File fh, MPI\_Offset offset, int whence)**

**• int MPI\_File\_read (MPI\_File fh, void \*buf, int count, MPI\_Datatype datatype, MPI\_Status \*status)**

**• int MPI\_File\_write (MPI\_File fh, const void \*buf, int count, MPI\_Datatype datatype, MPI\_Status \*status)**

**• int MPI\_File\_close (MPI\_File \*fh)**

\* Povratna vrednost svih funkcija je error code.

Otvaranje i zatvaranje fajla – one su grupne operacije

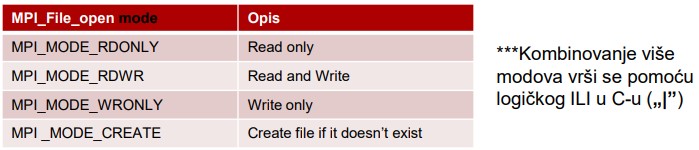
• Otvaranje fajla:

**• int MPI\_File\_open (MPI\_Comm comm, const char \*filename, int amode, MPI\_Info info, MPI\_File \*fh)**

• Comm – komunikator koji ukazuje na grupu procesa koja pristupa fajlu

• Filename – ime fajla

• Amode – mod za otvaranje fajla



• Info – omogućava prosleđivanje dodatnih parametara - hints – koristimo MPI\_INFO\_NULL

• Fh – povratna vrednost – pokazivač na otvoreni fajl

• Zatvaranje fajla

**• int MPI\_File\_close (MPI\_File \*fh)**

• Fh – pokazivač na fajl koji se zatvara

Upis i čitanje

• Čitanje iz fajla

**• int MPI\_File\_read (MPI\_File fh, void \*buf, int count, MPI\_Datatype datatype, MPI\_Status \*status)**

• Upis u fajl

**• int MPI\_File\_write (MPI\_File fh, const void \*buf, int count, MPI\_Datatype datatype, MPI\_Status \*status)**

• Fh - pokazivač na fajl

• Bafer u memoriji u koji se upisuje/iz kog se čita

• Count – broj podataka tipa datatype koji se čitaju/upisuju

• Status – funkcija ista kao u slučaju MPI\_Recv funkcije

Pozicioniranje

• Pozicioniranje unutar fajla

**• int MPI\_File\_seek (MPI\_File fh, MPI\_Offset offset, int whence)**

• Fh - pokazivač na fajl

• Offset - Pomeraj unutar fajla, podrazumevano u bajtovima

• Whence – u odnosu na šta se postavlja pokazivač

• MPI\_SEEK\_SET – Pointer se postavlja na vrednost pomeraja (od početka fajla)

• MPI\_SEEK\_CUR – Pointer se postavlja na trenutnu poziciju pokazivača + pomeraj

• MPI\_SEEK\_END - Pointer se postavlja na kraj fajla + pomeraj

• Primer:

• MPI\_File\_seek (fh, 0, MPI\_SEEK\_SET); - Pozicioniranje na početak fajla

• MPI\_File\_seek (fh, rank \* bufsize , MPI\_SEEK\_SET); - Pozicioniranje u zavisnosti od ranga procesa

Pre čitanja mora da se pozicioniramo!!!

Upotreba eksplicitnog pomeraja

• U prethodnom primeru korišćeni su individualni pointeri, koji koriste trenutnu lokaciju pokazivača kao mesto odakle se vrši čitanje/upis

• Drugi tip funkcija – funkcije sa eksplicitnim pomerajem (explicit-offset functions)

• **int MPI\_File\_read\_at (MPI\_File fh, MPI\_Offset offset, void \*buf, int count, MPI\_Datatype datatype, MPI\_Status \*status)** //file read i file seek

**• int MPI\_File\_write\_at (MPI\_File fh, MPI\_Offset offset, const void \*buf, int count, MPI\_Datatype datatype, MPI\_Status \*status)** //file write i file seek

• Funkcijama se direktno prosleđuje pomeraj (offset), ostali parametri su isti kao i u MPI\_File\_read i MPI\_File\_write funkcijama

Offset je u Bajtovima !!!

Nekontinualni pristup

• File view - Pogled na fajl

• Definiše koji deo fajla je „vidljiv“ od strane procesa

• Funkcije za čitanje/upis podataka mogu pristupati samo tom delu fajla (svi ostali podaci se preskaču)

• Podrazumevano, pri prvom otvaranju fajla, proces može pristupiti celom fajlu. Inicijalno, pomeraj (offset) je postavljen na 0!

• Funkcija za postavljanje pogleda:

**• int MPI\_File\_set\_view (MPI\_File fh, MPI\_Offset disp, MPI\_Datatype etype, MPI\_Datatype filetype, const char \* datarep, MPI\_Info info)**

• Disp – broj bajtova koji se preskače od početka fajla

• Etype – Tip podataka (može biti osnovni ili izvedeni tip). Predstavlja osnovnu jedinicu pristupa podacima. Svi pomeraji se specificiraju u odnosu na broj etype jedinica

• Filetype – Tip podatka (može biti osnovni ili izvedeni tip). Mora biti ili istog tipa kao etype, ili izvedeni tip koji se sastoji od više jedinica tipa etype

• Datarep (data representation) – podrazumevana vrednost native – reprezentacija podataka u fajlu je ista kao u memoriji

• Info – MPI\_INFO\_NULL (hints)

• Pogled na fajl počinje od zadatog pomeraja i sadrži više kontinualnih „kopija“ filetype-a

• Podrazumevani file view: disp = 0, filetype = MPI\_BYTE

• Funkcija se može pozivati i pogled na fajl se može menjati više puta u toku programa

# Grupne I/O operacije

• Grupne operacije za upis i čitanje podataka:

**• int MPI\_File\_read\_all (MPI\_File fh, void \*buf, int count, MPI\_Datatype datatype, MPI\_Status \*status)**

**• int MPI\_File\_write\_all (MPI\_File fh, const void \*buf, int count, MPI\_Datatype datatype, MPI\_Status \*status)**

• Parametri funkcija su isti kao u slučaju osnovnih MPI\_File\_read i MPI\_File\_write funkcija

Read ne zna za poglede i izvedene tipove

Kad svi procesi istog komunikatora vrše čitanje, onda read all

# Pristup nizovima u datotekama

**• int MPI\_Type\_create\_darray (int size, int rank, int ndims, int array\_of\_gsizes[], int array\_of\_distribs[], int array\_of\_dargs[], int array\_of\_psizes[], int order, MPI\_Datatype oldtype, MPI\_Datatype \*newtype)**

• Funkcija kreira izvedeni tip podatka opisom N-dimenzionalnog podpolja unutar N-dimenzionalnog polja

Na osnovu procesa MPI sam određuje offsete, tj. koji deo polja se šalje kom procesu.

• Size – Broj procesa među kojima je niz distribuiran

• Ndims – Broj dimenzija polja i podpolja (N)

• gsizes - Broj elemenata starog tipa (oldtype) u svakoj dimenziji polja (niz pozitivnih celih brojeva)

• distribs – Način distribucije - MPI\_DISTRIBUTE\_BLOCK

• dargs – Parametar distribucije za svaku dimenziju - MPI\_DISTRIBUTE\_DFLT\_DARG

• psizes – Broj procesa u svakoj dimenziji među kojima je polje distribuirano. Uzima se da grid procesa ima isti broj dimenzija kao polje i podpolje. Ako niz po nekoj dimenziji nije distribuiran, broj procesa u toj dimenziji je 1!

• Order - Način predstavljanja polja u memoriji, ili MPI\_ORDER\_C ili MPI\_ORDER\_FORTRAN

• Tip svakog elementa polja

• Treba voditi računa o deljivosti N i broja procesa.

Podrazumevano uzima se

**• int MPI\_Type\_create\_subarray (int ndims, int array\_of\_sizes[], int array\_of\_subsizes[], int array\_of\_starts, int order, MPI\_Datatype oldtype, MPI\_Datatype \*newtype)**

Eksplicitno se definiše koji deo polja ide kom procesu, mi samo definišemo offsete.

• Funkcija kreira izvedeni tip podatka definisanjem početka i dimenzija podpolja

• ndims - broj dimenzija polja (N)(pozitivan broj)

• sizes - broj elemenata starog tipa (oldtype) u svakoj dimenziji polja (niz pozitivnih celih brojeva)

• subsizes - broj elemenata starog tipa (oldtype) u svakoj dimenziji podpolja(niz pozitivnih celih brojeva)

• offsets - početne koordinate podpolja u svakoj dimenziji(niz nenegativnih brojeva)

• order - način predstavljanja polja u memoriji, ili MPI\_ORDER\_C ili MPI\_ORDER\_FORTRAN

# Neblokirajuće I/O funkcije

**• int MPI\_File\_iread (MPI\_File fh, void \*buf, int count, MPI\_Datatype datatype, MPI\_Request\* request)**

**• int MPI\_File\_iwrite (MPI\_File fh, ROMIO\_CONST void \*buf, int count, MPI\_Datatype datatype, MPI\_Request\* request)**

**• int MPI\_File\_iwrite\_at (MPI\_File fh, MPI\_Offset offset, const void \*buf, int count,MPI\_Datatype datatype, MPI\_Request\* request);**

**• int MPI\_File\_iread\_at (MPI\_File fh, MPI\_Offset offset, void \*buf, int count, MPI\_Datatype datatype, MPI\_Request\* request)**

**• int MPI\_Wait ( MPI\_Request \*request, MPI\_Status \*status );** - iz ove funkcije se proces vraća onda kada se operacija identifikovana sa request završi. Ova operacija je blokirajuća.

**• int MPI\_Test ( MPI\_Request \*request, int \*flag, MPI\_Status \*status );** - funkcija vraća informaciju o trenutnom stanju operacije koja je identifikovana argumentom request. Argument flag se posatavlja na true ukoliko je operacija završena, u suprotnom na false. Argument status sadrži dodatne statusne informacije. Ova operacija nije blokirajuća.

# Split collective I/O

• Grupne MPI I/O operacije obezbeđuju specifičnu formu neblokirajućih funkcija - Split collective I/O

• Kako bi se koristile grupne neblokirajuće I/O operacije, korisnik mora definisati početak i kraj operacije:

**• int MPI\_File\_write\_all\_begin (MPI\_File fh, const void \*buf, int count, MPI\_Datatype datatype);**

**• int MPI\_File\_write\_aIl\_end (MPI\_File fh, const void \*buf, MPI\_Status \*status);**

• Ograničenje – U jednom trenutku može biti aktivna samo jedna split collective I/O operacija nad jednim fajlom!

# Deljeni pokazivač na fajlove

• Pokazivač na fajl se može deliti između procesa u okviru komunikatora koji se prosleđuje prilikom otvaranja fajla. Redosled čitanja/upisa procesa nije važan.

• Funkcije za čitanje/upis sa trenutne pozicije deljenog pokazivača:

**• int MPI\_File\_write\_shared(MPI\_File fh, const void \*buf, int count, MPI\_Datatype datatype, MPI\_Status \*status);**

**• int MPI\_File\_read\_shared(MPI\_File fh, void \*buf, int count, MPI\_Datatype datatype, MPI\_Status \* status);**

**• int MPI\_File\_seek\_shared(MPI\_File fh, MPI\_Offset offset, int whence);**

• Neblokirajuća verzija funkcija sa deljenim pokazivačem

**• int MPI\_File\_iwrite\_shared(MPI\_File fh, ROMIO\_CONST void \*buf, int count, MPI\_Datatype datatype, MPI\_Request \*request)**

**• int MPI\_File\_iread\_shared(MPI\_File fh, void \*buf, int count, MPI\_Datatype datatype, MPI\_Request \*request)**

• Grupne operacije sa deljenim pokazivačem

**• int MPI\_File\_read\_ordered(MPI\_File fh, void \*buf, int count, MPI\_Datatype datatype, MPI\_Status \*status)**

**• int MPI\_File\_write\_ordered(MPI\_File fh, ROMIO\_CONST void \*buf, int count, MPI\_Datatype datatype, MPI\_Status \*status)**

• Upis i čitanje se u ovom slučaju vrše redom, po ranku procesa!