ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΑΡΑΛΛΗΛΟ ΥΠΟΛΟΓΙΣΜΟ - ΑΣΚΗΣΗ-Ι 2024-2025

<REDACTED>

Build and run

Για building και debugging του project χρησιμοποιήθηκε CMake, του οποίου το CMakeLists.txt συμπεριλαμβάνεται στα παραδοτέα αρχεία για λόγους αναφοράς, δεν είναι όμως σε καμία περίπτωση απαραίτητη η χρήση CMake για building του project.

Υπάρχουν δύο (εξίσου εύκολοι) τρόποι να γίνει compile το project:

1) Παρέχεται Makefile που το κάνει αυτόματα με τις εξής εντολές (από το root του project folder):

```
make
```

Υπάρχει και δυνατότητα διαγραφής των object files με:

```
make clean
```

2) Απευθείας compile με το gcc wrapper του MPI (mpicc) με:

```
mpicc ./src/*.c -I ./include -o bin
```

Εκτέλεση του προγράμματος γίνεται με (όπου <process_count> ακέραιο νούμερο των processes, χωρίς angled brackets, πχ. 2):

```
mpirun -np <process_count> bin
```

Ερώτηση 1

Ο κώδικας ξεκινάει με την υλοποίηση της λειτουργίας του κύριου process (0) το οποίο είναι υπεύθυνο για το διάβασμα του μήκους και των τιμών της ακολουθίας από τον χρήστη. Στο block που απεθύνεται σε process rank 0 γίνονται οι ακόλουθες λειτουργίες με την αντίστοιχη σειρά:

1) Κλήση του menu function που προβάλει το μενού επιλογών για τον χρήστη. Αυτό το μέρος καλύπτεται παρακάτω.

2) Ζητείται το απ'τον χρήστη το μήκος της ακολουθίας ακεραίων καθώς και οι αριθμοί που την απαρτίζουν (η συνάρτηση read_int έχει ενσωματομένο optional έλεγχο εγκυρότητας).

3) Οι παραπάνω δύο πληροφορίες αποστέλονται σε όλα τα άλλα processes (δηλαδή σε αυτά με μη μηδενικό rank) με δύο MPI_Send() εντός for loop.

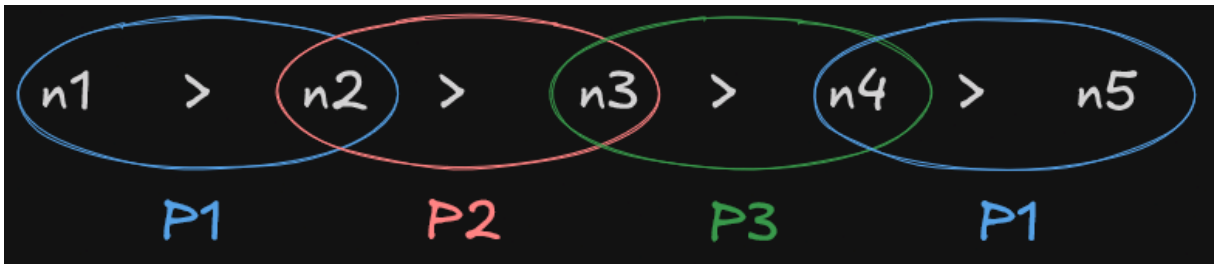
Ερώτηση 2

Η λογική των processes που κάνουν τις συγκρίσεις έχει ως εξής:

1) Το process λαμβάνει το μήκος της λίστας και τα στοιχεία της (τα οποία αποθηκεύονται) σε νέα δεσμευμένη (και εκμηδενισμένη λόγω του calloc) μνήμη.

2) Τα στοιχεία της ακολουθίας ελέγχονται ανά δύο μεταξύ τους για το αν ισχύει $n_i > n_{i+1}$. Σε περίπτωση που η συνθήκη είναι αληθής, ο δείκτης του στοιχείου μετά απ'το οποίο η ακολουθία παύει να είναι σε αύξουσα σειρά αποθηκεύεται και το πρόγραμμα βγαίνει εκτός επανάληψης με break. Η επανάληψη εντός της οποίας τρέχει η σύγκριση χρησιμοποιεί ως iterator το process rank (-1 καθώς για προσπέλαση μνήμης χρειαζόμαστε 0-based indexing) με την λογική πως δε γίνεται ισοκατανομή των αριθμών της ακολουθίας, αλλά των ζευγαριών αριθμών εντός της ακολουθίας που αντιστοιχούν σε μία σύγκριση το κάθε ένα. Συνεπώς κάθε process παίρνει ένα ζευγάρι αριθμών με "round robin μοτίβο".

```
for(int i = process_id - 1; i < sequence_length - 1; i +=
(process_count - 1)) {
    /* Check if ith element is greater than i+1th,
    and thus not in ascending order */
    if(array[i] > array[i + 1]) {
        order_breaks_after = i;
        break;
    }
}
```



3) Τέλος, ο δείκτης αποστέλεται πίσω στο process 0 και αποδεσμεύεται η μνήμη στην οποία είχε αποθηκευθεί η ακολουθία.

Ερώτηση 3

Το process 0 λαμβάνει πίσω τους δείκτες των στοιχείων μετά απ΄τα οποία "σπάει" η ακολουθία και τους αποθηκεύει σε μόλις δεσμευμένη συνεχόμενη μνήμη για να βρεθεί ο ελάχιστος. Καθώς η ακολουθία είναι γραμμική, αρκεί να προσδιοριστεί ο μικρότερος δείκτης στοιχείου στο οποίο "σπάει" για να δούμε που σταματάει η αύξουσα σειρά.

Extras

Το πρόγραμμα έχει μενού επιλογών, το οποίο εμφανίζεται στην πρώτη εκτέλεση, καθώς και μετά από κάθε εκτέλεση μέχρι ο χρήσης να δώσει την τιμή 2 για έξοδο απ'το πρόγραμμα. Επιπροσθέτως, οποιαδήποτε άλλη τιμή από τις επιτρεπτές (1 & 2) οδηγεί σε καθάρισμα του terminal και σε

επανεμφάνιση του menu. Αυτή η συμπεριφορά επιτυγχάνεται με ένα απλό goto label.

```
void menu() {
    invalid_option:

    // menu print, ommited to save space in the doc

    int option;
    read_int(&option);
    if((option != 1) && (option != 2)) {
        goto invalid_option;
    }

    if(option == 2) {
      /* Terminate program, MPI_finalize() hangs so abort
        is used, kind of a bodge */
      MPI_Abort(MPI_COMM_WORLD, 1);
      exit(0);
    }

  system("clear");
}
```

1) Το πρόγραμμα τρέχει εντός ατέρμονως βρόγχου, έτσι ο μόνος τρόπος τερματισμού (πέρα από Interrupt με ^C) είναι μέσω της κλήσης exit() που γίνεται μέσα στο menu function κατόπιν της επιλογής 2.

2) Γίνεται χρήση αυτοσχέδιας συνάρτησης εισόδου (ακεραίων) καθώς η scanf() δεν έχει error reporting για σφάλμα μετατροπής string.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "input_sanitizer.h"

#define BUFFER_SIZE 1024

/* Reads an integer from stdin with built-in, optional validity
check for non-ints and (also) optionally negative ints,
```

```
   safe(er maybe) replacement for scanf. */

/* int read_int(int* var, int validity_check_enabled, int
allow_negative)

where validity_check_enabled checks for the input's validity
if true and allow_negative (applicable only when
validity_check_enabled is true) specifies whether to let
negative integers pass the check or not */

// ReSharper disable once CppNotAllPathsReturnValue
int read_int(int* var, ...) {
    va_list args;
    va_start(args, var);

    char buffer[BUFFER_SIZE];

    // If read fails, stop
    if(!fgets(buffer, BUFFER_SIZE, stdin))
       return EXIT_FAILURE;

    int validity_check_enabled = va_arg(args, int);
    if(validity_check_enabled) {
        int allow_negative = va_arg(args, int);
        char *str = "n";

        if (!allow_negative)
            str = " non-negative";

        va_end(args);

        while (!input_sanitizer(buffer, allow_negative)) {
            printf("\n\033[3;31mInvalid, must be a%s integer!
            Retry...\033[0m\n", str);
            fgets(buffer, BUFFER_SIZE, stdin);
        }
    }

    const int val = (int) strtol(buffer, NULL, 10);
    *var = val;
```

```
    return EXIT_SUCCESS;
}
```

3) Ενδεικτικά τρεξίματα:

```
Input the length of the sequence:
4

Input number (1/4):
1

Input number (2/4):
2

Input number (3/4):
3

Input number (4/4):
4

(Yes) Sequence is in ascending order.

Press any key to continue...
```

```
Input the length of the sequence:
4

Input number (1/4):
3

Input number (2/4):
4

Input number (3/4):
5
```

```
Input number (4/4):
1

(No) Order breaks after element 3.

Press any key to continue...
```