

# Εργασία 4η

Ομάδα 10

Αξελού Ολυμπία 2161

Τσιτσοπούλου Ειρήνη 2203

# Παραδοχές εργασίας

- 1) Σε περίπτωση οποιουδήποτε συντακτικού λάθους, στο 4.1 τερματίζει το πρόγραμμα, και στο 4.2 τερματίζει όλο το περιβάλλον εκτέλεσης
- 2) Αν γίνει ένα run (non\_existing\_file), η εντολή run αγνοείται και ο χρήστης μπορεί να δώσει άλλη εντολή
- 3) Ένα πρόγραμμα που έχει τελειώσει παραμένει στη λίστα με τις αρχικοποιημένες τιμές (πχ core = -1 και running = 0), δεν το λαμβάνουμε υπόψη στο rebalancing, το κάνει free η main τερματίζοντας
- 4) Σε μεταβλητή τύπου \$arg[\$k] όπου \$arg[ ] είναι global πίνακας, \$k είναι local μεταβλητή
- 5) Θεωρούμε ότι ο χρήστης θα δώσει enter μετά την εντολή RETURN
- 6) Υλοποίηση σηματοφόρων: με pthread mutexes (assignment 2)
- 7) Η εναλλαγή προγραμμάτων σε κάθε core γίνεται ανά TIME\_TO\_SWITCH εντολές
- 8) Οι λίστες globals & locals είναι ίδες όσον αφορά τα nodes τους (var\_storage.c), ο συγχρονισμός της globals γίνεται κατά την κλήση των συναρτήσεων με κλείδωμα όλης της λίστας
- 9) Ομοίως και για την λίστα των προγραμμάτων: κλείδωμα όλης της λίστας για την προσπέλασή της

# Βελτιστοποιήσεις

- 1) Labels: Με τη βοήθεια της λίστας, σε κάθε jump προς τα πάνω (σε εντολές από τις οποίες έχει περάσει) πάει αμέσως χωρίς να ξαναψάχνει όλο το αρχείο (περίπτωση jump προς τα κάτω)
- 2) Εξισορρόπηση φόρτου των cores :
  - a) Στην αρχή του τρεξίματος νέου προγράμματος με την ανάθεσή του στο core με τα λιγότερα προγράμματα
  - b) Με το πέρας ενός προγράμματος: rebalancing στους κόμβους (Εύρεση προγραμμάτων ενεργών και προγραμμάτων μπλοκαρισμένων και κατανομή αυτών στα cores)

# 4.1 libraries

**var\_storage.h:** αποθήκευση καθολικών και τοπικών μεταβλητών

```
typedef volatile struct variable{
    char *name;
    int value;
    volatile struct variable * next;
    volatile struct variable * prev;
}varT;

//var_storage.h functions
destroy_list, abort_function, init_list, print_contents, add_node, find_name,
find_array_name, create_array, realloc_array, modify_node, read_node
```

**labels.h:** αποθήκευση labels εντολών για εύρεσή τους από εντολές branch

```
typedef struct label_struct{
    char *name;
    off_t offset;
    struct label_struct *prev;
    struct label_struct *next;
}labelsT;

//labels.h functions
destroy_labels, init_labels, print_labels, add_label, search_label
```

**program\_handler1.h:** αποθήκευση στοιχείων του προγράμματος που τρέχει (στο 4.1 μόνο ένα πρόγραμμα τη φορά)

*#include "var\_storage.h" #include "labels.h"*

```
typedef volatile struct program_struct{
    char *name;
    int fd;
    varT *locals;
    labelsT *labels;
}programT;
```

**parser.h:** ανάγνωση input file, διερμηνεία, εκτέλεση εντολών σε C

*#include "program\_handler.h"*

# realloc\_array & create\_array

## var\_storage.c

//array[k] is given and array does not exist: create positions 0->k in array

```
varT *create_array(varT *head, char array_name[ ], int new_last_index){
    int i;
    char name[NAME_SIZE];
    varT* last_array_cell;

    for(i = 0; i <= new_last_index; i++){
        sprintf(name, "%s[%d]", array_name, i);
        last_array_cell = add_node(head, head->prev, name);           //προσθήκη κόμβων στο τέλος της λίστας
    }
    return last_array_cell;
}
```

//array[k] is given and array[n] exists with n<k: create positions n+1->k in array

```
varT *realloc_array(varT *head, varT *current, char array_name[ ], int old_last_index, int new_last_index){
    int i;
    char name[NAME_SIZE];
    varT* last_array_cell;

    for(i = old_last_index+1; i <= new_last_index; i++, current = current->next){
        sprintf(name, "%s[%d]", array_name, i);
        last_array_cell = add_node(head, current, name);             //προσθήκη κόμβων μετά το τελευταίο στοιχείο του πίνακα
    }
    return last_array_cell;
}
```

καλούνται από την find\_array\_name

# Εύρεση labels στις εντολές branch

Εντολές Branch:

Αρχικά ψάχνει στη λίστα (search label)

Αν βρει το label στην λίστα -> loop (επειδή έχει ήδη διατρέξει την εντολή)

Αλλιώς, ψάχνει στο αρχείο (search\_label\_downwards) στις επόμενες εντολές

Διαβάζοντας το αρχείο, τη στιγμή που βρίσκουμε ένα label, το αποθηκεύουμε στην λίστα, πριν διαβάσουμε την εντολή της ίδιας γραμμής

```
off_t search_label(labelsT *head, char name[], int print_flag){
    labelsT *current;

    for (current = head->next; current->name != NULL; current = current->next){
        if(strcmp(name, current->name) == 0){
            //Found label with (name)
            return(current->offset);
        }
    }
    //did not found label with (name)
    return(INVALID_OFFSET);
}
```

```
void search_label_downwards(int fd, char temp_char, char label_to_find[], labelsT *labels){
    int i, bytes_read; char input_buffer[LABEL_SIZE];

    while (1) {
        //discard '\n'
        bytes_read = my_read(fd, input_buffer, __LINE__);
        if (bytes_read == 0) //EOF: label not found (error)

        else if(input_buffer[0] == '\n'){
            //discard_spaces and '\n'
            if (strcmp(label_to_find, input_buffer)){
                if(is_command(input_buffer) == 0){ //found another label (not label_to_find) - add to list
                    add_label(labels, input_buffer, lseek(fd, 0, SEEK_CUR) - strlen(input_buffer) - 1);
                }

                continue; //go to next line
            }
            //label_to_find has been found
            //main will jump to label_to_find and read label from file (offset does not need to change)
            break;
        }
    }
}
```

## 4.2 libraries

**var\_storage.h:** αποθήκευση καθολικών και τοπικών μεταβλητών

```
typedef volatile struct variable{
    char *name;
    int value;
    volatile struct variable * next;
    volatile struct variable * prev;
}varT;
```

//var\_storage.h functions

destroy\_list, abort\_function, init\_list, print\_contents, add\_node, find\_name,  
find\_array\_name, create\_array, realloc\_array, modify\_node, read\_node

**labels.h:** αποθήκευση labels εντολών για εύρεσή τους από εντολές branch

```
typedef struct label_struct{
    char *name;
    off_t offset;
    struct label_struct *prev;
    struct label_struct *next;
}labelsT;
```

//labels.h functions

destroy\_labels, init\_labels, print\_labels, add\_label, search\_label

**program\_handler2.h:** αποθήκευση στοιχείων του προγράμματος που τρέχει (στο 4.1 μόνο ένα πρόγραμμα τη φορά)

*#include "var\_storage.h" #include "labels.h"*

```
typedef volatile struct program_struct{
    char *name;
    int core, id, running, blocked, sleep_time, fd, woke_up, running_now;
    time_t sleep_start;
    varT *locals, *down_sem;
    labelsT *labels;
    volatile struct program_struct *next,*prev;
}programT;
```

//program\_handler2.h functions

init\_program\_list, add\_program,search\_program\_id  
destroy\_programs, remove\_program  
print\_programs, find\_less\_busy\_core, rebalance\_cores

**parser.h:** ανάγνωση input file, διερμηνεία, εκτέλεση εντολών σε C

*#include "program\_handler.h"*

Main – Core – user\_interface : threads & synchronization & Globals' declaration

```
pthread_mutex_t glob_mtx (declared in parser.h) , program_mtx
program *programs
varT *globals (declared in parser.h)
volatile int exit_given, nofcores
```

main

core

## user\_interface

# Core thread, sleep implementation and rebalancing cores

```
CORE

// take arguments
program_counter = 0;
while(1){
    if(exit_given) { break; }

    mtx_lock(&program_mtx);
    if(program_counter == TIME_TO_SWITCH || current_program->core != my_no){
        program_counter = 0; current_program = current_program->next;
    }

    if(current_program->core == my_no){
        program_counter++;
        if(current_program->running){

            if(!current_program->blocked){
                current_program->running_now = 1;
                mtx_unlock(&program_mtx);
                main_program(current_program, programs, &program_mtx);
                mtx_lock(&program_mtx);
                current_program->running_now = 0;

                if(current_program->running == 0){
                    //program returned
                    //destroy labels and local variables list
                    current_program->core = INIT_CORE; close(current_program->fd)
                    rebalance_cores(programs, nofcores);
                    current_program = programs;
                }
            }
            else if(current_program->blocked){
                if(current_program->sleep_start >= 0){
                    if(time(NULL) - current_program->sleep_start >= current_program->sleep_time){
                        current_program->blocked = 0;
                        current_program->sleep_start = current_program->sleep_time = -1;
                    }
                }
                else if(current_program->down_sem != NULL){ // blocked by down (προηγούμενο slide)}
                else{ //kill: same as return }
            }
        }
        mtx_unlock(&program_mtx, __LINE__);
    }
}
```

thread safe: Ανάλογα με την εντολή το main\_program (parser.c) αν χρειαστεί κλειδώνει την λίστα globals & την λίστα programs (πχ UP)

εναλλαγή προγραμμάτων ανά TIME\_TO\_SWITCH εντολές (ή TIME\_TO\_SWITCH προσπάθειες για εκτέλεση εντολών)

## Program\_handler2

### // Rebalancing

```
running_core = blocked_core = 0;

for(current: all nodes in list of programs){
    if(current->running && !current->running_now){
        if(current->blocked)
            current->core = (blocked_core++) % nofcores;
        else
            current->core = (running_core++) % nofcores;
    }
}
```

## PARSER

```
//RETURN
program->running = 0;
```

```
// SLEEP
//read variable or value = varval1

program->blocked = 1;
program->sleep_start = time(NULL);
program->sleep_time = varval1;
//will not wake up before (current) time + varval1
}
```



# UP/DOWN in core thread and in parser.c

## CORE

```
while(1){
...
//if current_program is blocked and not by sleep
if(current_program->down_sem != NULL){
    if(current_program->woke_up){ //sb woke me up
        current_program->blocked = 0;
        current_program->down_sem = NULL;
        current_program->woke_up = 0;
    }
}
...
}
```

## PARSER

```
// DOWN
//find semaphore name in list

mtx_lock(&glob_mtx);
//find node of semaphore in list = current
mtx_unlock(&glob_mtx);

varval1 = varval1 - 1;
if (varval1 < 0){
    mtx_lock(program_mtx);
    program->down_sem = current; //points at the sem I blocked at
    program->blocked = 1;
    mtx_unlock(program_mtx);
}

mtx_lock(&glob_mtx);
modify_node(globals, varval_op, varval1);
mtx_unlock(&glob_mtx);
}
```

## PARSER

```
// UP
//find global variable name in list

mtx_lock(&glob_mtx);
//find node of global variable in list = current
mtx_unlock(&glob_mtx);

varval1 = varval1 - 1;
if (varval1 <= 0){
    mtx_lock(program_mtx);
    for(program_current : all nodes in programs list){
        if(program_current->blocked && program_current->down_sem == current){
            program_current->woke_up = 1; //inform core I woke up someone blocked at sem
            break;
        }
    }
    mtx_unlock(program_mtx);
}
mtx_lock(&glob_mtx);
modify_node(globals, varval_op, varval1);
mtx_unlock(&glob_mtx);
}
```

# Globals' List: LOAD/STORE

## // STORE

## PARSER

```
// read variable names : global_var & local_var

localVarValue = read_varval(in locals, local_var, ...) // in parser.c

mtx_lock(&glob_mtx);
modify_node(globals, global_var, varval1);      // in var_storage.c
mtx_unlock(&glob_mtx);
}
```

## // LOAD

## PARSER

```
// read variable name : local_var

// read variable name : global_var
varval1 = check_varGlobal(in globals & locals (in case of array), global_var) // in parser.c

modify_node(locals, local_var, varval1); //in var_storage.c
```

## // check\_varGlobal

## PARSER

```
if //check if variable (eg. $var)
    mtx_lock(&glob_mtx);
    value = read_node(globals, input_buffer, !PRINT_REPORT);
    mtx_unlock(&glob_mtx);
    return value;
}
else{
    if(strchr(var_name, '$')){ // check if simple array (eg. $var[1])
        mtx_lock(&glob_mtx, __LINE__);
        value = read_node(globals, input_buffer, !PRINT_REPORT);
        mtx_unlock(&glob_mtx, __LINE__);
        return value;
    }
    else{// it's array with local var for position (eg. $var[$temp])
        // read value of local variable (eg of $temp) (no synchronization)
        // and change var_name string (eg $var[$temp] -> $var[2])
        // and search for it to read its value

        mtx_lock(&glob_mtx);
        value = read_node(globals, input_buffer, !PRINT_REPORT);
        mtx_unlock(&glob_mtx);

        return value;
    }
}
```

