Ταυτόχρονος Προγραμματισμός

Σαράφογλου Δημήτρης

Ρούμπος Γιάννης

1.1 Fifo pipe

Πίνακες για γράψιμο και διάβασμα από και προς τα αρχεία

```
typedef struct read_args {
    char *array;
    char *array2;
} READ_ARGS;
typedef struct write_args {
    char *array;
    char *array2;
} WRITE_ARGS
```

Οι πληροφορίες για το ring buffer και επιπλέον πληροφορίες

```
typedef struct pipe {
    char c;
    struct pipe *next;
} PIPE;

typedef struct pipe_t {
    PIPE *pipe_root;
    PIPE *write_ptr;
    PIPE *read_ptr;
    int isEmpty;
    int write_access;
    int ID;
} PIPE_T;
```

Main thread

{
-Αρχικοποίηση των στοιχείων του struct
-Δημιουργία των pipes
-Δημιουργία των threads(1 για το διαβασμα και 1 για το γραψιμο)
-Περιμένει τα threads να τερματισουν
-Ελευθερώνει την αρχικα δεσμευμένη μνήμη
-Τερματίζει
}

Write thread

{
Μέχρι να φτάσουμε στο τέλος του αρχείου
{
Περίμενε μεχρι να είναι σειρά να γράψει
γράφουμε στο pipe ακολουθώντας την πολιτική ring buffer
Εκτυπώνουμε τι γράψαμε για επαλήθευση
Ειδοποιώ ότι δεν θέλω άλλο να γράψω
}

‘Οσο το pipe είναι άδειο
{
γράφουμε στο pipe ακολουθώντας την πολιτική ring buffer
Εκτυπώνουμε τι γράψαμε για επαλήθευση
Ειδοποιώ ότι δεν θέλω άλλο να γράψω
}
Ειδοποιώ ότι τελείωσαμε το γράψιμο
}

Read thread

{
Μέχρι το pipe να είναι αδειο κανε nop();
Μέχρι διαβάσει τα δεδομένα
{
Περίμενε μεχρι να θέλει να διαβάσει
Διάβασε & εκτύπωσε για δικιο σου ελεγχο
Ειδοποιήσε οτι δεν θες να διαβάσεις
}
Αντέγραψε τον buffer στο αρχείο
Κλείσε το αγωγό για γράψιμο
Μέχρι να διαβάσει τα δεδομένα
{
Περίμενε μέχρι να θέλει να διαβάσει
Διάβασε & εκτύπωσε για δικιο σου ελεγχο
Ειδοποιήσε οτι δεν θες να διαβάσεις
}
Ειδοποιησε ότι τελείωσες το γράψιμο
}

1.2 Prime Numbers with Multiple Threads

- Αρχικά ορίσαμε έναν τύπο δεδομένων με στοιχεία τον αριθμό προς υπολογισμό και τις πιθανές καταστάσεις που μπορεί να βρεθεί ένα νήμα δηλαδή αν το νήμα είναι διαθέσιμο ή όχι και αν είναι η σειρά του.
- Στην δημιουργία των νημάτων(pthread_create) περνάμε σαν μεταβλητές τις διευθύνσεις τους και εναν δείκτη προς αυτα με πληροφορίες τον παραπάνω τύπο δεδομένων.
- Έπειτα εφόσον διαβάσουμε έναν αριθμό βρίσκουμε τον πρώτο διαθέσιμο νήμα και αναθέτουμε τον αριθμό και ειδοποιούμε ότι το νήμα δεν είναι πλεον διαθέσιμο
- Έτσι, αφού περιμένουμε όλα τα νήματα να γινουν διαθέσιμα και να τερματίσουν φεύγουμε από το main νήμα

- Ότι αφορά την υλοποιήση του νήματος(-ων) όταν ειδοποιηθούμε από το main thread ότι μπορούμε να υπολογίσουμε την τιμή που μας έχει ανατεθεί και είμαστε ελεύθεροι και υπολογίζουμε το αποτέλεσμα μέσω κατάλληλης συνάρτησης και το εκτυπώνουμε.
- Τέλος, αλλάζουμε την κατάσταση του νήματος σε ελεύθερο και εφόσον εχουμε την κατάλληλη ειδοποιήση απο την main μπορούμε να τερματίσουμε.

Το struct που χρησιμοποιήσαμε:

```
typedef struct thread_info {
    int number;
    int signal; /* 0: worker not available,
1: worker available,
2: termination signal from main,
3: termination signal from worker. */
} THREAD_INFO;
```

1.3 External Merge Sort

Main thread

Υπολογίζει το size του file που δίνουμε ως όρισμα
Ελεγχει αν μπορει να γίνει απλη merge sort
Αν δεν μπορει δημιουργεί τα threads
Αρχικοποιεί τις αρχικές συνθήκες του struct
Περιμένει τα thread να τελειώσουν
Και συγχωνεύει τα 2 δυο αρχεία

Work thread

Υπολογίζει εκ νεου τα καινουργια sizes και ορίζει δείκτες στην αρχή κάθε αρχείου
Ελέγχει το μέγεθος των αρχείων και εκτελεί απλή merge sort(αν size<64b)
Αλλιώς δημιουργεί δυναμικά καινούργια 2 threads ώστε αναδρομικά να κάνει την παραπάνω διαδικασία
Περιμένει τα δυναμικά δεσμευμένα threads να τελειώσουν και
Συγχωνεύει τα αρχεία που δημιουργήθηκαν

Αναλυτικότερα για την λογική που ακολουθήσαμε και κάποιες συναρτήσεις.

Η λογική είναι ότι καθώς ανεβαίνουμε τα στάδια της αναδρομής δηλαδή αφου έχουμε φτάσει στο σημείο για την ταξινόμηση των δεδομένων μεσω απλου merge sort να δημιουργούμε ενα προσωρινό αρχείο και να συγχωνεύουμε τα 2 ηδη ταξινομημένα. Με αυτόν τον τρόπο ελέγχουμε καλύτερα την ροή των δεδομένων μεταξύ των αρχείων κατα την ταξινόμηση.

Οι συναρτήσεις του απλού merge sort ήταν γνωστές και χρησιμοποιήθηκαν ενώ αυτή του merging δυο αρχείων υλοποιήθηκε στη λογική ότι αφού και τα δύο αρχεία είναι ηδη ταξινομημένα αρκεί να συγκρίνουμε ένα προς ένα τα byte και να τα αποθηκεύσουμε στο τελικό αρχείο.

Τέλος να σημειώσουμε ότι δεν υπήρχαν συνθήκες ανταγωνισμού μόνο την αναμονή της main μεχρι να τελειώσουν την αναδρομή τα threads αλλά και την αναμονή των threads μέχρι να τελειώσουν τα δυναμικά δεσμευμένα «παιδιά τους»