ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Εργασία 3η

Προγραμματισμός Συστήματος

Μαγδαληνή Κορδορούμπα
ΑΜ: 1115201000203
Περίοδος 2013-2014

Αρχεία

Server:

- server.c
- s_functions.c
- s_functions.h
- queue.c
- queue.h
- Makefile

Client:

- client.c

Κοινά αρχεία:

- Makefile

Run

Τα ορίσματα των αρχείων είναι αυτά που ορίζονται στην εκφώνηση της άσκησης, και μπορούν να δοθούν με οποιαδήποτε σειρά.

Δημιουργήθηκαν επίσης δύο scripts, ένα για τον server και ένα για τους client (script_server.sh και script_client.sh). Το πρώτο τρέχοντάς το θα εκτελέσει τον server και μετά από ένα sleep 5, θα στείλει ένα σήμα SIGTERM, ενώ το δεύτερο θα εκτελέσει 4 clients (για κάθε ένα θα φτιάξει ένα ξεχωριστό φάκελο).

Server

Η δομή του Server είναι η παρακάτω:

```
pthread mutex_ουράς queue_lock
pthread cond insert              /* Όταν η ουρά είναι γεμάτη σταματάει το calling thread */
pthread cond worker_cond         /* Όταν η ουρά είναι άδεια σταματάει το calling thread */

main {
        δημιουργία των worker threads
        δημιουργία socket

        /* Με SIGTERM κάνει αποδεύσμευση πόρων και τερματίζει */
        while (δεν_έχουμε_λάβει_SIGTERM ){
                Περιμένει για συνδέσεις
                Δημιουργεί νέο detached client thread για κάθε σύνδεση
        }
}
```

client_thread {

- κάνει detach στον εαυτό του
- διαβάζει το όνομα του φακέλου που ζητάει ο πελάτης
- φτιάχνει δείκτη σε int, δεσμεύει δυναμικά μνήμη, και τον αρχικοποιεί με τον αριθμό των αρχείων που θα σταλθούν
  (ο δείκτης σε αυτόν τον μετρητή αποθηκεύεται σε όλους τους κόμβους της ουράς που αφορούν τον ίδιο πελάτη)
- φτιάχνει δείκτη σε pthread mutex, δεσμεύει δυναμικά μνήμη και τον αρχικοποιεί
  (ο δείκτης σε αυτόν τον mutex αποθηκεύεται σε όλους τους κόμβους της ουράς που αφορούν τον ίδιο πελάτη)
- βρίσκει τα αρχεία που θέλει να στείλει
- πριν εισάγει στην ουρά κάνει lock mutex(queue_lock)
- σε περίπτωση που η ουρά είναι γεμάτη, περιμένει: pthread_cond_wait (&insert , &queue_lock)
- όταν η ουρά αδειάσει κάνει insert στην ουρά
- στέλνει σήμα σε κάποιο worker thread που είναι ανενεργό, ενημερώνοντάς το ότι η ουρά δεν είναι πλέον άδεια pthread_cond_signal(&worker_cond)
- κάνει unlock mutex(queue_lock)

}

worker_thread {

while (δεν_έχει_ληφθεί_SIGTERM) {

- κάνει lock mutex(queue_lock)
- αν η ουρά είναι άδεια περιμένει
  pthread_cond_wait (&worker_cond , &queue_lock);
- όταν έρθει σήμα ότι η ουρά δεν είναι άδεια, αφαιρεί τον πρώτο κόμβο της ουράς, παίρνοντας έτσι ένα αρχείο για αντιγραφή
- στέλνει σήμα σε κάποιο client thread που είναι ανενεργό, ενημερώνοντάς το ότι η ουρά δεν είναι πλέον γεμάτη pthread_cond_signal(&worker_cond)
- κάνει unlock mutex(queue_lock)

- κλειδώνει τον κοινό mutex των κόμβων με ίδια socket
- στέλνει το αρχείο και όλες του τις πληροφορίες
- μειώνει το μετρητή των αρχείων που μένουν να σταλθούν στη συγκεκριμένη socket
- ξεκλειδώνει τον κοινό mutex
- Αν όλα τα αρχεία αυτής της Socket έχουν σταλθεί, κάνει αποδέσμευση του mutex και του counter.

}

}

Client

Ο Client έχει πιο απλή δομή.

Δημιουργεί σύνδεση με τον server.
Στέλνει το όνομα του καταλόγου που θέλει.
Διαβάζει και δημιουργεί τα αρχεία (και τους καταλόγους που χρειάζονται μέσω της mkdir()) που του έρχονται από το server, μέχρι ο server να ενημερώσει για το ότι τα αρχεία τελείωσαν.

Πηγές Κώδικα

Για τη σύνδεση του client και του server μέσω socket, χρησιμοποιήθηκε κώδικας από τα "int_str_server.c" και " int_str_client.c" που υπάρχουν στη σελίδα του μαθήματος, σε συνδυασμό με τις διαφάνειες του μαθήματος.

Η write_all και η read_all επίσης βασίστηκαν στις διαφάνειες του μαθήματος.

Οι συναρτήσεις της queue, είχαν υλοποιηθεί από εμένα για τις ανάγκες της προηγούμενης εργασίας του μαθήματος. Χρειάστηκαν κάποιες αλλαγές φυσικά, αλλά σε γενικές γραμμές ήταν ίδιες.

Η color_text βρέθηκε από το διαδίκτυο, από ένα forum για προγραμματισμό:
http://cboard.cprogramming.com/c-programming/75659-adding-color-c-program.html#post536331
Και κατανοήθηκε και ολοκληρώθηκε μέσω της Wikipedia.
http://en.wikipedia.org/wiki/ANSI_escape_code

Θα ήθελα εδώ να πω, ότι ο χρωματισμός της εξόδου της άσκησης έγινε καθαρά για λόγους ευκολότερης αναγνωσιμότητας. Σε περίπτωση που σας φαίνεται δύσχρηστο, θα μπορούσατε να αλλάξετε το σχολιασμό στην αρχή του αρχείου server.c:

Είναι ως εξής:
```
int worker_color = BLACK;
int client_color = CYAN;
int beg_end_color = RED;

/*
int worker_color = -1;
int client_color = -1;
int beg_end_color = -1;
*/
```

Και εσείς μπορείτε να το αλλάξετε σε:
```
/*
```

```
int worker_color = BLACK;
int client_color = CYAN;
int beg_end_color = RED;
*/
int worker_color = -1;
int client_color = -1;
int beg_end_color = -1;
```