Περιγραφή άσκησης 3

Η άσκηση αυτή αποτελείται από 3 τμήματα, εκ των οποίων το πρώτο είναι γραμμένο σε κέλυφος bash και τα άλλα δύο σε C.

Το πρώτο μέρος, ο WebCreator, δημιουργεί w\*p σελίδες html, σύμφωνα με τον αλγόριθμο που έχει δοθεί στην εκφώνηση της άσκησης. Κάθε τέτοια σελίδα έχει ως source file το βιβλίο που προτείνεται σαν link στην εκφώνηση, ενώ αποτελείται από m γραμμές και f+q συνδέσμους. Αν ο χρήστης δώσει είσοδο w=2 και p=2, τότε είναι σίγουρο ότι το αρχείο όλες οι σελίδες θα έχουν τουλάχιστον έναν εισερχόμενο σύνδεσμο. Η επαλήθευση αυτή γίνεται με τη σύγκριση των unique ονομάτων όλων των συνδέσμων που υπάρχουν στα αρχεία με το σύνολο των συνδέσμων που δημιουργήθηκαν τυχαία. Αν τα δύο αυτά σύνολα είναι ίδια, τότε υπάρχει ένας τουλάχιστον εισερχόμενος σύνδεσμος για κάθε σελίδα, αλλιώς όχι.

Το δεύτερο μέρος, ο WebServer, αποτελείται από 4 ομάδες νημάτων. Το ένα νήμα είναι η main που δημιουργεί όλα τα ακόλουθα νήματα. Το δεύτερο νήμα είναι ο producer, ο οποίος «υποδέχεται» κάθε πελάτη στην πόρτα -p και τοποθετεί το socket του πελάτη σε μια κοινόχρηστη ουρά. Στην ουρά αυτή έχουν πρόσβαση οι consumers (τρίτη ομάδα νημάτων), οι οποίοι πρακτικά εξυπηρετούν τον εκάστοτε πελάτη και τερματίζουν αμέσως μετά. Τέλος, υπάρχει και το νήμα commander που εξυπηρετεί πελάτες (που έρχονται κυρίως μέσω telnet) στην πόρτα -c. Στην περίπτωση SHUTDOWN, στέλνεται από τον commander ειδικό μήνυμα τερματισμού στους consumers, ενώ ο producer τερματίζεται με pthread\_cancel.

Ο WebCrawler αποτελείται από 3 ομάδες νημάτων, δηλαδή τη main, τον producer και τους consumers. Αυτή τη φορά, ο producer τοποθετεί το αρχικό url στην ουρά και στη συνέχεια αναλαμβάνει το ρόλο του JobExecutor της άσκησης 2, δημιουργώντας πάντοτε 2 Workers με τα αντίστοιχα named pipes. Και εδώ, ο τερματισμός γίνεται με ομαλό τρόπο όπως και στο παραπάνω ερώτημα. Οι consumers, αναλαμβάνουν το ρόλο του «κατεβάσματος» της σελίδας από τον WebServer και στη συνέχεια τη σκανάρουν για άλλα links. Αν αυτά δεν υπάρχουν ήδη στο δίσκο και δε βρίσκονται στην ουρά, τότε τοποθετούνται στην ουρά. Κατά την απομάκρυνση στοιχείου από την ουρά, δημιουργείται τότε (ενώ ο mutex είναι ακόμα κλειδωμένος) ένα «άδειο» αρχείο, ώστε αν τύχει κατά το κατέβασμα και ένα άλλο νήμα βρει link για το ίδιο αρχείο, να μην το ξανατοποθετήσει στην ουρά. Έτσι, αφού το thread «κατεβάσει» το περιεχόμενο του (ως τώρα) άδειου αρχείου, το ανοίγει και το συμπληρώνει.

Η άσκηση ελέγχθηκε μέσω valgrind και δεν υπάρχει κανένα memory leak τόσο στον WebServer όσο και στον WebCrawler. Επίσης, εμπεριέχεται και Makefile που υλοποιεί separate compilation.

Τόσο στον WebServer όσο και στον WebCrawler, για κάθε κοινόχρηστη συνθήκη ελέγχου (π.χ. πλήρης buffer ή empty buffer) έχω βάλει αντίστοιχη μεταβλητή συνθήκης, ώστε να μην υπάρχει καθόλου busy waiting. Αντίστοιχα έχω βάλει και mutexes για την προστασία των κοινόχρηστων πόρων. Για τη υλοποίηση του WebServer και του WebCrawler, χρησιμοποίησα κώδικα από τις διαφάνειες του μαθήματος (κώδικας για server/client σε tcp sockets και κώδικας παραγωγού καταναλωτή με κυκλικό buffer).

Ενδεικτικές εκτελέσεις

[εδώ θα βάλεις μερικές εκτελέσεις]