Εισαγωγή

Στόχος αυτής της εργασίας είναι να εξοικειωθείτε με τη δημιουργία διεργασιών (processes) χρησιμοποιώντας τα system calls fork/exec, την επικοινωνία διεργασιών μέσω pipes, τη χρήση low-level I/O και τη δημιουργία bash scripts.
Στα πλαίσια αυτής της εργασίας θα υλοποιήσετε μία απλουστευμένη έκδοση του dropbox όπου διαφορετικές διεργασίες θα πρέπει να συγχρονίσουν ένα σύνολο από αρχεία. Οι διεργασίες λειτουργούν όλες με τον ίδιο τρόπο και ο στόχος τους είναι να επικοινωνήσουν και να συγχρονιστούν μεταξύ τους με στόχο να έχουν όλες από ένα κοινό αντίγραφο από τα αρχεία.

A) Η εφαρμογή mirror_client (70%)

Σε αυτή την εργασία θα υλοποιήσετε μία εφαρμογή την mirror_client η οποία θα χρησιμοποιείται ως εξής:

```
./mirror_client -n id -c common_dir -i input_dir -m mirror_dir -b buffer_size -l log_file
```

- `id`: είναι ένας αριθμός που αποτελεί ένα αναγνωριστικό για το συγκεκριμένο client
- `common_dir`: είναι ένα directory το οποίο θα χρησιμοποιηθεί για επικοινωνία μεταξύ των mirror_clients.
- `input_dir`: είναι ένα directory το οποίο περιέχει τα αρχεία τα οποία συνεισφέρει ο συγκεκριμένος client.
- `b`: είναι το μέγεθος του buffer για διάβασμα πάνω από τα pipes.
- `mirror_dir`: είναι ένα directory το οποίο περιέχει τα αρχεία από όλους τους client που συμμετέχουν στο mirroring. Αυτό το directory το φτιάχνει κάθε client ξεχωριστά για να αποθηκεύσει όλα τα αρχεία που θα λάβει από τους άλλους clients.
- `log_file`: εκεί γράφονται όλα τα μηνύματα κατά τη διάρκεια εκτέλεσης του client

Παράδειγμα εκτέλεσης:

```
./mirror_client -n 1 -c ./common -i ./1_input -m ./1_mirror -b 100 -l log_file1
./mirror_client -n 2 -c ./common -i ./2_input -m ./2_mirror -b 200 -l log_file2
```

Ξεκινάμε δύο clients με τα ids 1 και 2, θα χρησιμοποιήσουν το ./common directory για επικοινωνία με τους άλλους clients και το directory 1_input και 2_input αντίστοιχα για τα αρχεία που συνεισφέρουν και 1_mirror και 2_mirror για να αποθηκεύσουν τα αρχεία που λαμβάνουν. Σκοπός είναι οι clients να αρχίσουν να επικοινωνούν αυτόνομα ώστε μετά από κάποια χρονική στιγμή όλοι να έχουν ένα αντίγραφο από όλα τα αρχεία.
Η λειτουργία κάθε client είναι η εξής:

1. Όταν ξεκινάει ελέγχει τις παραμέτρους εισόδου και συγκεκριμένα αν υπάρχουν τα directories. Αν δεν υπάρχει το input_dir τότε ο client τερματίζει με μήνυμα λάθους. Αν το directory mirror_dir υπάρχει τότε επίσης τερματίζει με μήνυμα λάθους γιατί σημαίνει πως κάποιος άλλος client το έχει δημιουργήσει και το χρησιμοποιεί. Αν το directory common_dir και το mirror_dir δεν υπάρχουν τα δημιουργεί.
2. Ο client πηγαίνει και γράφει στο directory common ένα αρχείο ίδιο με το id του με την κατάληξη id. Μέσα στο αρχείο γράφει το process_id του. Π.χ. Στο παραπάνω παράδειγμα το αρχείο ./common/1.id θα περιέχει το pid 1234 και το ./common/2.id θα περιέχει το 1235. Αν το αρχείο αυτό υπάρχει τότε ο client τερματίζει με λάθος καθώς σημαίνει πως υπάρχει κάποιος client που τρέχει με το ίδιο id.
3. Ο client περιοδικά ελέγχει το common_dir για όλα τα αρχεία. Κατά τη διάρκεια εκτέλεσης, αρχεία *.id μπορεί να εμφανιστούν (επειδή καινούργιοι clients εισέρχονται) ή να διαγραφούν (βλέπε παρακάτω).
4. **Για κάθε καινούργιο id** που βλέπει ο client και που δεν έχει επεξεργαστεί κάνει τα εξής:

a. Κάνει fork δύο διεργασίες παιδιά που θα αναλάβουν να συγχρονιστούν με τα αντίστοιχα δύο παιδιά από τους άλλους clients. Το ένα παιδί είναι για τη λήψη αρχείων και το άλλο για την αποστολή. Τα παιδιά δημιουργούν από ένα named pipe στο directory ./common με το εξής φορμάτ:
common_dir/id1_to_id2.fifo και common_dir/id2_to_id1.fifo. Αν τα pipes υπάρχουν σημαίνει πως η άλλη διαδικασία τα έχει φτιάξει και περιμένει οπότε τα ανοίγουμε.

b. Τα δύο αυτά pipes θα χρησιμοποιηθούν για ανταλλαγή πληροφοριών με το εξής πρωτόκολλο:
   i. Ένα process με id1, για κάθε αρχείο στο directory id1_input και για κάθε άλλο process id2,
      1. Γράφει στο pipe id1_to_id2.fifo 2 bytes που δηλώνουν το μήκος του ονόματος του αρχείου
      2. Γράφει στο pipe id1_to_id2.fifo το όνομα του αρχείου
      3. Γράφει στο pipe id1_to_id2.fifo 4 bytes που δηλώνουν το μήκος του αρχείου
      4. Γράφει στο pipe id1_to_id2.fifo το αρχείο
   ii. Όταν τελειώσει με τα αρχεία στέλνει τα 2 πρώτα bytes να είναι 00.
   iii. Αντίστοιχα συμβαίνει και το ανάποδο, δηλαδή το process με το id2 στέλνει τα αρχεία του στο pipe id2_to_id1.
   iv. Για κάθε αρχείο που λαμβάνεται τυπώνεται αντίστοιχο μήνυμα με το όνομα και τα bytes που διαβάστηκαν στο log_file.
   v. Κάθε αρχείο που λαμβάνεται αποθηκεύεται στο mirror_directory με το εξής format: mirror_dir/id/filename.
   vi. Αν ολοκληρωθεί επιτυχώς η μεταφορά (φτάσουμε δηλαδή να διαβάσουμε 00) ο πατέρας το ανιχνεύει (π.χ. Μέσω της wait()) και τυπώνει ανάλογο μήνυμα.
   vii. Αν υπάρξει λάθος κατά τη μεταφορά, το παιδί επίσης τερματίζει και στέλνει αντίστοιχο signal (SIGUSR) στον πατέρα. Ο πατέρας τυπώνει ανάλογο μήνυμα και ξαναδοκιμάζει να κάνει fork τα αντίστοιχα παιδιά για να ξαναγίνει η μεταφορά από την αρχή, μέχρι όμως 3 φορές. Μετά την 3η φορά τα παρατάει οριστικά.
   viii. Αν ένα παιδί περιμένει πάνω από 30 seconds να διαβάσει από το pipe χωρίς να ξεκινήσει κάποια είσοδος τότε τερματίζει, τυπώνει το λάθος και στέλνει signal (SIGUSR) στον πατέρα.
   ix. Το αρχείο μπορεί να περιέχει και subdirectories τα οποία θα πρέπει να δημιουργηθούν αντίστοιχα μέσα στην ιεραρχία στο mirror_dir. Για παράδειγμα ένα όνομα αρχείου του client 5 μπορεί να είναι tmp/one/two.txt. Αυτό θα πρέπει να δημιουργηθεί στο mirror_dir/5/tmp/one/two.txt.

5. Στην περίπτωση που ένα id εξαφανιστεί από το directory τότε ο κάθε client κάνει fork ένα παιδί που αναλαμβάνει να σβήσει τα αντίστοιχα αρχεία από το τοπικό mirror directory. Για παράδειγμα αν το id 5 εξαφανιστεί τότε όλοι οι clients θα πρέπει να σβήσουν όλο το directory mirror/5/.
6. Όταν ένας client πάρει ένα signal SIGINT ή SIGQUIT τότε κάνει τα εξής:
   a. Σβήνει όλο το τοπικό του mirror_dir
   b. Σβήνει το id file από το common_dir

B) Το script create_infiles.sh (20%)

Σε αυτό το script ο στόχος σας είναι να δημιουργήσετε μια ιεραρχία με αρχεία και directories που θα χρησιμοποιηθεί σαν είσοδο από κάθε client. Φυσικά κατά τη διάρκεια της ανάπτυξης της εφαρμογής στο Α) μπορείτε να χρησιμοποιήσετε λίγα και μικρά αρχεία για να κάνετε debug. Το script create_infiles.sh δουλεύει ως εξής:

```
./create_infiles.sh dir_name num_of_files num_of_dirs levels
```

- `dir_name`: το directory όπου θα γραφτούν τα αρχεία.

- `num_of_files`: το συνολικό πλήθος των αρχείων
- `num_of_dirs`: το συνολικό πλήθος των directories
- `levels`: σε πόσα επίπεδα θα κατανέμονται τα directories

Το script κάνει τα εξής:

1. Κάνει ελέγχους για νούμερα εισόδου
2. Αν το dir_name δεν υπάρχει το δημιουργεί
3. Δημιουργεί τα directory names με τυχαία αλφαριθμητικά μήκους από 1 έως 8 χαρακτήρες και γράμματα (π.χ. 138kjd ή jkkm180k)
4. Ξεκινάει και φτιάχνει τα directories επίπεδο-επίπεδο μέχρι να φτιάξει num_of_dirs directories σε levels επίπεδα. Για παράδειγμα αν έχουμε τα ονόματα aaa, bbb, cc, dddd, eee (num_of_dirs=5) και levels=2 τότε θα φτιάξουμε με τη σειρά:
    a. dir_name/aaa/
    b. dir_name/aaa/bbb
    c. dir_name/cc/
    d. dir_name/cc/dddd
    e. dir_name/eee
5. Αντίστοιχα δημιουργείτε num_of_files τυχαία ονόματα αρχείων από 1 έως 8 χαρακτήρες. Τα αρχεία αυτά τα κατανέμετε ομοιόμορφα στα directories που δημιουργήσατε με round-robin σειρά. Για παράδειγμα αν έχουμε τα ονόματα f1, f2, f3 … f10 τότε η τελική ιεραρχία θα έχει ώς εξής:
    a. dir_name/f1
    b. dir_name/f7
    c. dir_name/aaa/f2
    d. dir_name/aaa/f8
    e. dir_name/aaa/bbb/f3
    f. dir_name/aaa/bbb/f9
    g. dir_name/cc/f4
    h. dir_name/cc/f10
    i. dir_name/cc/dddd/f5
    j. dir_name/eee/f6
6. Σε καθένα από αυτά τα αρχεία θα γράψετε ένα τυχαίο αλφαριθμητικό από 1 kb έως 128 kb.

C) To script get_stats.sh (10%)

Ο στόχος αυτού του script είναι να επεξεργαστεί τα logfiles από όλους τους clients και να βγάλει στατιστικά. Το script θα τρέχει με τη μορφή:

cat log_file1 log_file2 … log_filen | ./get_stats.sh και θα τυπώνει:

- Πόσοι clients συνδέθηκαν και ποιοι (μία λίστα με όλα τα ids από τους clients)
- Το ελάχιστο και το μέγιστο id των clients
- Το πλήθος των bytes που στάλθηκαν
- Το πλήθος bytes που διαβάστηκαν
- Το πλήθος των αρχείων που στάλθηκαν
- Το πλήθος των αρχείων που λήφθηκαν
- Το πλήθος των clients που έφυγαν από το σύστημα

D) Encryption bonus (20%)

Στην άσκηση αυτή μπορείτε (ως bonus) να υλοποιήσετε τη μεταφορά των αρχείων με encryption κάνοντας χρήση του gpg για τη δημιουργία public keys, για encryption και για decryption. Σε αυτή την περίπτωση στο αρχείο id θα πρέπει να βάλετε το public key του κάθε client. Το key αυτο θα

χρησιμοποιείται για το encryption των περιεχομένων του αρχείου. Όταν το αρχείο ληφθεί από την άλλη πλευρά θα πρέπει να αποθηκεύεται decrypted.

Παρατηρήσεις

- Η συγκεκριμένη εργασία απαιτεί αρκετή σκέψη και καλό σχεδιασμό όσον αφορά κατανεμημένους πόρους, forks, blocking/non-blocking I/O κλπ. Η άσκηση δεν περιγράφει όλες τις λεπτομέρειες και τις δομές για τον απλό λόγο πως οι σχεδιαστικές επιλογές είναι αποκλειστικά δικές σας (βεβαιωθείτε φυσικά πως τις περιγράφετε αναλυτικά στο README). Αν έχετε διάφορες επιλογές για κάποιο σημείο της άσκησης σκεφτείτε τα υπέρ και τα κατά, τεκμηριώστε τα στο README, επιλέξτε αυτό που θεωρείτε σωστό και λογικό και περιγράψτε γιατί το επιλέξατε στο README.

Παραδοτέα

- Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματός σας. 1-2 σελίδες ASCII κειμένου είναι αρκετές. Συμπεριλάβετε την εξήγηση και τις οδηγίες για το compilation και την εκτέλεση του προγράμματός σας σε ένα αρχείο README μαζί με τον κώδικα που θα υποβάλετε.
- Οποιαδήποτε πηγή πληροφορίας, συμπεριλαμβανομένου και κώδικα που μπορεί να βρήκατε στο Διαδίκτυο ή αλλού θα πρέπει να αναφερθεί στον πηγαίο κώδικά σας αλλά και στο παραπάνω README.
- Όλη η δουλειά σας (πηγαίος κώδικας, Makefile και README) σε ένα tar.gz file με ονομασία OnomaEponymoProject2.tar.gz. Προσοχή να υποβάλετε μόνο κώδικα, Makefile, README και όχι τα binaries.

Διαδικαστικά

- Για επιπρόσθετες ανακοινώσεις, παρακολουθείτε το forum του μαθήματος στο piazza.com. Η πλήρης διεύθυνση είναι https://piazza.com/uoa.gr/spring2019/k24/home. Η παρακολούθηση του φόρουμ στο Piazza είναι υποχρεωτική.
- Το πρόγραμμά σας θα πρέπει να γραφεί σε C (ή C++). Στην περίπτωσή που χρησιμοποιήσετε C++ δεν μπορείτε να χρησιμοποιήσετε τις έτοιμες δομές της Standard Template Library (STL). Σε κάθε περίπτωση το πρόγραμμά σας θα πρέπει να τρέχει στα Linux workstations του Τμήματος.
- Ο κώδικάς σας θα πρέπει να αποτελείται από τουλάχιστον δύο (και κατά προτίμηση περισσότερα) διαφορετικά αρχεία. Η χρήση του separate compilation είναι επιτακτική και ο κώδικάς σας θα πρέπει να έχει ένα Makefile.
- Βεβαιωθείτε πως ακολουθείτε καλές πρακτικές software engineering κατά την υλοποίηση της άσκησης. Η οργάνωση, η αναγνωσιμότητα και η ύπαρξη σχολίων στον κώδικα αποτελούν κομμάτι της βαθμολογίας σας.

Άλλες σημαντικές παρατηρήσεις

- Οι εργασίες είναι ατομικές.
- Όποιος υποβάλει / παρουσιάσει κώδικα που δεν έχει γραφτεί από την ίδια/τον ίδιο μηδενίζεται στο μάθημα.
- Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πώς θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιασδήποτε μορφής) είναι κάτι που δεν επιτρέπεται. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικα απλά παίρνει μηδέν στο μάθημα. Αυτό ισχύει για όσους εμπλέκονται ανεξάρτητα από το ποιος έδωσε/πήρε κλπ.
- Οι ασκήσεις προγραμματισμού μπορούν να δοθούν με καθυστέρηση το πολύ 3 ημερών και με ποινή 5% για κάθε μέρα αργοπορίας. Πέραν των 3 αυτών ημερών, δεν μπορούν να κατατεθούν ασκήσεις.