Εργαστηριακή Άσκηση

Χρονοπρογραμματιστής Διεργασιών

Εισαγωγή

Το ζητούμενο στην εργασία είναι η υλοποίηση ενός περιβάλλοντος χρονοπρογραμματισμού σε λειτουργικό σύστημα Unix. Συγκεκριμένα, πρέπει να υλοποιηθεί ένας δρομολογητής ο οποίος παίρνει σαν είσοδο τις εφαρμογές που πρέπει να εκτελέσει, διαβάζοντας ένα αρχείο με τα ονόματα τους, τις εισάγει σε μία κατάλληλη δομή δεδομένων (λίστα) και κατόπιν τις δρομολογεί εφαρμόζοντας μία από τις ακόλουθες πολιτικές:

- **Εξυπηρέτηση με βάση τη σειρά άφιξης (FCFS)**: οι εφαρμογές εκτελούνται με τη σειρά εισαγωγής τους στη λίστα, δηλαδή με τη σειρά που αναγράφονται στο αρχείο εισόδου
- **Εξυπηρέτηση με βάση τη μικρότερη διάρκεια (SJF)**: κάθε εφαρμογή χαρακτηρίζεται από έναν στατικό χρόνο εκτέλεσης (αριθμό), ο οποίος καθορίζει τη σειρά εκτέλεσής της. Οι εφαρμογές με το μικρότερο αριθμό έχουν τη μεγαλύτερη προτεραιότητα. Αν δύο εφαρμογές έχουν τον ίδιο αριθμό τότε προτεραιότητα έχει η διεργασία που εμφανίζεται πρώτη στο αρχείο εισόδου.
- **Δρομολόγηση εκ περιτροπής (RR)**: οι εφαρμογές εκτελούνται εκ περιτροπής για χρόνο ίσο με προκαθορισμένο κβάντο δρομολόγησης.
- **Δρομολόγηση με βάση την προτεραιότητα (PRIO)**: κάθε εφαρμογή χαρακτηρίζεται από μια **στατική** προτεραιότητα (αριθμό). Οι εφαρμογές με το μικρότερο αριθμό έχουν μεγαλύτερη προτεραιότητα και εκτελούνται πρώτα. Μεταξύ εφαρμογών με τον ίδιο βαθμό προτεραιότητας, εφαρμόζεται δρομολόγηση εκ περιτροπής.

Η υλοποίηση του δρομολογητή πρέπει να πραγματοποιηθεί με τέτοιο τρόπο ώστε οι βασικές δομές δεδομένων και οι εσωτερικές του λειτουργίες να είναι όσο το δυνατόν περισσότερο συμπαγείς και ανεξάρτητες της πολιτικής που εφαρμόζει. Π.χ., η πολιτική θα μπορούσε να μεταβάλλεται δυναμικά κατά τον χρόνο εκτέλεσης. Η παρούσα ενότητα αυτή περιλαμβάνει μια γενική περιγραφή των βασικών στοιχείων της υλοποίησης του δρομολογητή που θα πρέπει να λάβετε υπόψη.

Περιγραφέας εφαρμογής

Κάθε εφαρμογή που πρόκειται να εκτελεστεί από τον δρομολογητή θα περιγράφεται με μία κατάλληλη δομή δεδομένων. Μία τέτοια δομή θα δεσμεύεται και αρχικοποιείται κατάλληλα για κάθε εφαρμογή που διαβάζεται από το αρχείο εισόδου και πρόκειται να εκτελεστεί από το δρομολογητή. Στην συνέχεια θα εισάγεται στην ουρά εκτέλεσης. Η δομή αυτή θα περιλαμβάνει πεδία που περιγράφουν χαρακτηριστικά της εφαρμογής, όπως π.χ. όνομα εκτελέσιμου αρχείου, προτεραιότητα, αναγνωριστικό (pid), κατάσταση εκτέλεσης κ.α.

Δυνατές καταστάσεις εκτέλεσης μιας εφαρμογής (διεργασίας) είναι οι εξής:

- **READY**: Η εφαρμογή έχει μόλις εισαχθεί σε ουρά για εκτέλεση. Οι εφαρμογές εκτελούνται μέσω του δρομολογητή, ο οποίος δημιουργεί τη διεργασία για την εκτέλεση μιας εφαρμογής (*με fork-exec*) .
- **RUNNING**: Η εφαρμογή – διεργασία είναι ενεργή.
- **STOPPED**: Η εκτέλεση της εφαρμογής έχει διακοπεί.
- **EXITED**: Η εφαρμογή έχει τερματίσει.

Ουρά Εκτέλεσης

Η δομή δεδομένων που περιέχει τους περιγραφείς των εφαρμογών πρέπει να είναι μία διπλά συνδεδεμένη λίστα (ουρά). Εισαγωγή διεργασιών πραγματοποιείται στο τέλος της λίστας ενώ εξαγωγή από την αρχή της. Στην περίπτωση της δρομολόγησης με βάση την προτεραιότητα, μπορείτε να χρησιμοποιήσετε τόσες λίστες όσες και οι δυνατές τιμές προτεραιοτήτων (π.χ. κάποιος μικρός ακέραιος) ή να ακολουθήσετε κάποια άλλη προσέγγιση.

Δρομολόγηση Εφαρμογών

Ο δρομολογητής επιλέγει κάθε φορά την επόμενη προς εκτέλεση εφαρμογή, και εφόσον υπάρχει κάποια εφαρμογή πραγματοποιεί τις κατάλληλες ενέργειες διαφορετικά τερματίζει.

Θεωρούμε ότι οι εφαρμογές που θα δρομολογηθούν περιλαμβάνουν αποκλειστικά υπολογισμούς και καθόλου εντολές εισόδου – εξόδου.

Κατά συνέπεια, στην περίπτωση των στατικών πολιτικών δρομολόγησης (FCFS, SJF), ο δρομολογητής θα ενεργοποιείται μόνο όταν μια εφαρμογή – διεργασία τερματίζει.

Αντίθετα, στην περίπτωση των δυναμικών πολιτικών δρομολόγησης (RR, PRIO), ο δρομολογητής θα ενεργοποιείται κατά τακτά χρονικά διαστήματα, όπως καθορίζεται από το κβάντο δρομολόγησης που αποτελεί παράμετρο του προγράμματος. Ο μηχανισμός ενεργοποίησης του δρομολογητή θα υλοποιηθεί με χρήση κατάλληλου timer (π.χ. με χρήση της κλήσης **nanosleep**).

Στην κατηγορία των δυναμικών πολιτικών δρομολόγησης, η διαχείριση της εκτέλεσης των εφαρμογών – διεργασιών, δηλαδή η αναστολή και η συνέχιση της εκτέλεσής τους, θα πραγματοποιείται με χρήση των σημάτων **SIGSTOP** και **SIGCONT**.

Τέλος, όταν μια εφαρμογή – διεργασία τερματίζει, ο δρομολογητής θα πρέπει να ενημερώνεται, έχοντας θέσει χειριστή για το σήμα **SIGCHLD,** και να εκτελεί τις απαραίτητες ενέργειες για τη σωστή λειτουργία του.

Σε κάθε περίπτωση, όταν μια εφαρμογή τερματίζει θα πρέπει να υπολογίζεται και να εκτυπώνεται ο συνολικός χρόνος εκτέλεσής της, ξεκινώντας από τη στιγμή που μπήκε στην λίστα εκτέλεσης, αλλά και ο χρόνος που έχει περάσει από την εκκίνηση του δρομολογητή. Για λόγους απλότητας, όλες οι διεργασίες

Χρήσιμες υποδείξεις

1. Ο δρομολογητής θα πρέπει να λαμβάνει το σήμα **SIGCHLD** μόνο όταν μια εφαρμογή – παιδί τερματίζει και όχι όταν αναστέλλεται η εκτέλεσή του (βλ. **sigaction**)
2. Είναι χρήσιμο ο δρομολογητής να διατηρεί ιστορικό εκτέλεσης, δηλαδή πληροφορία με την τρέχουσα εκτελούμενη εφαρμογή σε κάθε κβάντο.
3. Κατά τον τερματισμό μιας διεργασίας, ο δρομολογητής θα πρέπει να εμφανίζει πληροφορίες για την εκτέλεση της αντίστοιχης εφαρμογής. Εναλλακτικά, η εκτύπωση των πληροφοριών μπορεί να πραγματοποιηθεί πριν την ολοκλήρωση της εκτέλεσης του δρομολογητή. Ο δρομολογητής θα πρέπει να εμφανίζει και το συνολικό χρόνο εκτέλεσης του συνόλου των εφαρμογών.
4. Για την ανάπτυξη του δρομολογητή, μπορείτε να χρησιμοποιήστε απλά προγράμματα που επιτρέπουν τον έλεγχο της σωστής λειτουργίας του.
5. Ο κώδικας των παραπάνω προγραμμάτων, χαρακτηριστικά αρχεία εισόδου, ένα αρχείο (script) με παραδείγματα εκτέλεσης του δρομολογητή, ένα ενδεικτικό παράδειγμα εκτύπωσης αποτελεσμάτων, καθώς και ένας ενδεικτικός κώδικας του δρομολογητή, τον οποίο πρέπει να συμπληρώσετε, δίνονται μαζί με την εκφώνηση της άσκησης.

Χρήση

Σύμφωνα με τα παραπάνω, το πρόγραμμα του δρομολογητή θα χρησιμοποιείται ως εξής: **scheduler <policy> [<quantum>] <input_filename>,** όπου

- **scheduler:** το εκτελέσιμο του δρομολογητή που θα υλοποιήσετε.
- **policy:** η πολιτική δρομολόγησης με την οποία θα εκτελεστούν οι εφαρμογές. Δυνατές τιμές είναι οι FCFS, SJF, RR, PRIO.
- **quantum:** Το κβάντο δρομολόγησης σε msec. Απαιτείται μόνο όταν ως πολιτική δρομολόγησης έχει οριστεί μία από τις RR ή PRIO.
- **input_filename:** Το όνομα του αρχείου το οποίο περιέχει το φορτίο εργασιών που θα εκτελεστούν μέσω του δρομολογητή. Κάθε γραμμή του αρχείου περιλαμβάνει το όνομα του εκτελέσιμου μιας εφαρμογής και έναν αριθμό που θα δηλώνει τον απαιτούμενο χρόνο εκτέλεσης (σύμφωνα με κάποια σχετική ή απόλυτη κλίμακα) ή την προτεραιότητα του.

Παραδείγματα χρήσης:

$ scheduler FCFS reverse.txt

$ scheduler SJF reverse.txt

$ scheduler RR 1000 reverse.txt

$ scheduler PRIO 500 reverse.txt

Καλείστε να συμπληρώστε το ζητούμενο πρόγραμμα δρομολόγησης το εκτελέσιμο του οποίου θα ονομάσετε scheduler.

Αναλυτικότερα:

1. Υλοποιήστε τις βασικές δομές και λειτουργίες του χρονοδρομολογητή.
2. Υλοποιήστε την πολιτική δρομολόγησης FCFS.
3. Υλοποιήστε την πολιτική δρομολόγησης SJF.
4. Υλοποιήστε την πολιτική δρομολόγησης RR.
5. Υλοποιήστε την πολιτική δρομολόγησης PRIO.
6. Εκτελέστε και αξιολογήστε τις παραπάνω πολιτικές για τα παραδείγματα εκτέλεσης του δρομολογητή όπως ορίζονται στο αρχείο run.sh.
7. Γράψτε την αναφορά σύμφωνα με τις οδηγίες και παρουσιάστε τα αποτελέσματα από τα πειράματά σας.

Θα βαθμολογηθείτε ξεχωριστά για όλα τα παραπάνω ζητούμενα. Αν δεν έχετε υλοποιήσει όλες τις πολιτικές δρομολόγησης, τότε φροντίστε ώστε ο κώδικας σας να εκτυπώνει κατάλληλο μήνυμα για αυτές αλλά και να εκτελείται σωστά για όλες τις υπόλοιπες πολιτικές.

Παρατηρήσεις:

- Μην συμπεριλάβετε εκτελέσιμα αρχεία στο αρχείο zip με τον κώδικα σας.
- Μπορείτε να κάνετε οποιεσδήποτε αλλαγές στον ημιτελή κώδικα του δρομολογητή αλλά θα πρέπει να ακολουθήσετε τη διεπαφή (δηλαδή τις παραμέτρους εκτέλεσης) που ορίζονται στην εκφώνηση και χρησιμοποιούνται στα παραδείγματα εκτέλεσης.
- Ο κώδικας που θα παραδώσετε θα πρέπει να μπορεί να μεταγλωττιστεί και να εκτελέσει scripts αντίστοιχα με το run.sh. Το πρώτο στάδιο αξιολόγησης του κώδικα της εργασίας σας θα πραγματοποιηθεί με αυτό ακριβώς τον τρόπο.
- Τα εκτελέσιμα για τα βοηθητικά προγράμματα και τον δρομολογητή μπορούν να δημιουργηθούν με την εκτέλεση της εντολής make στο τερματικό και ενώ είστε στο αντίστοιχο directory.
- Ο χρόνο εκτέλεσης των βοηθητικών προγραμμάτων (load1, load2, …) μπορεί να προσαρμοστεί θέτοντας κατάλληλη τιμή στη σταθερά DELAY, όπως φαίνεται και στο αρχείο Makefile.
- Μια υλοποίηση της εργασίας έχει υλοποιηθεί και δοκιμαστεί σε συστήματα Linux (όπως και στο σύστημα diogenis) χωρίς προβλήματα.
- Αν εκτελέσετε τα πειράματά σας σε σύστημα το οποίο χρησιμοποιείται από ταυτόχρονα από πολλούς χρήστες (π.χ. diogenis), τότε ο χρόνος εκτέλεσης των βοηθητικών προγραμμάτων αναμένεται να διαφέρει κάθε φορά, ανάλογα με το συνολικό φόρτο του συστήματος.

Καλή επιτυχία!