Λειτουργικά Συστήματα

2025 - 2026

2η Εργαστηριακή Άσκηση

Σκοπός

Σκοπός της εξαμηνιαίας εργαστηριακής άσκησης (project) είναι να γνωρίσουν, να κατανοήσουν και να εξοικειωθούν οι φοιτητές με όσα θέματα πραγματεύεται το μάθημα και να αναγνωρίσουν τις δυνατότητες της συνεργασίας σε ομάδες. Τις ομάδες θα έχετε την ευκαιρία να τις επιλέξετε εσείς και να τις οργανώσετε όπως θέλετε. Η εμπειρία εργασίας σε ομάδες και η συνεργασία είναι σημαντική δεξιότητα που δεν πρέπει να αμελήσετε να καλλιεργήσετε γιατί, όπως θα διαπιστώσετε ως απόφοιτοι, είναι σημαντικό προσόν ενός μηχανικού.

Ομάδα Project

Οι ομάδες project θα πρέπει να αποτελούνται από 1 έως 4 άτομα, και προτείνεται να είναι ίδια η σύνθεση των ομάδων στις δύο ασκήσεις.

Παραδοτέα Project

Τα παραδοτέα κάθε ομάδας είναι:

- Η αναφορά του project της ομάδας, σε μορφή pdf.
- Τα αρχεία του κώδικα της υλοποίησης σε συμπιεσμένη μορφή zip. **Στην αρχή του αρχείου του κώδικα, σε σχόλια, θα τοποθετείτε τα ονοματεπώνυμα και τους ΑΜ των μελών της ομάδας ως εξής:**

```
/* Spiridon Sioutas, 2345678 */
/* Christos Makris, 4567890 */
/* Panagiotis Hadjidoukas, 3456789 */
/* Aristidis Ilias, 1234567 */
```

Τα παραδοτέα θα υποβληθούν μέσω του eclass στις «Εργασίες», από το ίδιο μέλος της ομάδας που είχε παραδώσει τα παραδοτέα του πρώτου project.

Θα παραδοθούν:

- η αναφορά σε pdf στην εργασία «2ο project 2025-2026, Αναφορά Ομάδων»
- τα αρχεία του κώδικα σας σε μορφή zip στην εργασία «2ο project 2025-2026, Κώδικας Ομάδων»

Στο περιεχόμενο της αναφοράς της κάθε ομάδας, η τεκμηρίωση είναι απαραίτητη και θα αξιολογηθεί. Στο εξώφυλλο της αναφοράς, πρέπει να περιλαμβάνει τα ονόματα των μελών της ομάδας, τους αριθμούς μητρώου και το email τους.

Θα αποτελείται από μια σύντομη περιγραφή του σχεδιασμού της υλοποίησης κάθε ομάδας, χρησιμοποιώντας διαγράμματα η/και ψευδοκώδικα όπου θεωρείτε απαραίτητο.

Επίσης, θα περιγράφει εν συντομία τα προβλήματα που αντιμετωπίσατε κατά την υλοποίηση της άσκησης και τις προσεγγίσεις της ομάδας για την επίλυση τους. Εάν εντοπίσατε περισσότερες από μια προσεγγίσεις για την επίλυση, να τις αναφέρετε και να τεκμηριώσετε γιατί επιλέξατε όποια επιλέξατε.

Ειδικά και επιπρόσθετα, στις ασκήσεις που πρέπει να αναπτύξετε κάποιας μορφής κώδικα/πρόγραμμα, ο κώδικας σας πρέπει να απαντά ακριβώς σε ό,τι σας ζητείτε, χωρίς αφαιρέσεις ή προσθήκες, π.χ., αν ζητείται να υλοποιηθεί κάτι με διεργασίες (processes), αλλά επειδή δυσκολευτήκατε ή δεν καταλάβατε κάτι καλά, το υλοποιήσατε με νήματα (threads), δεν θα πάρετε τα credits που αναλογούν. Οι απαντήσεις που αφορούν κώδικα μέσα στο pdf της αναφοράς θα πρέπει να τεκμηριώνονται με screenshots από την εκτέλεση των προγραμμάτων.

Στην αναφορά (pdf), στο μέρος που αντιστοιχεί στις υλοποιήσεις σας, πρέπει να αναφέρει στην αρχή της τεκμηρίωσης:

- ποια τμήματα της άσκησης έχετε υλοποιήσει και ποια όχι
- ποια δουλεύουν σωστά και ποια δεν δουλεύουν.

ΜΗΝ συμπεριλάβετε στην αναφορά σας την εκφώνηση της άσκησης και τον κώδικά σας αυτούσιο, αλλά καλείστε να περιγράψετε λεκτικά τον αλγόριθμο, κώδικα και προσέγγιση που ακολουθήσατε.

Επικοινωνία και παράδοση του project

Όλη η επικοινωνία του μαθήματος και κατά συνέπεια και όλη η επικοινωνία για το project και η παράδοση του project θα γίνεται αποκλειστικά μέσω του eClass του μαθήματος.

Οι απορίες θα συζητούνται **πρωτίστως στο μάθημα** και δευτερευόντως με μήνυμα («Μηνύματα» του eclass) μέσω του eClass.

Η παράδοση θα γίνει μέσω eClass, μέσα από τις «Εργασίες» του μαθήματος.

Απορίες με e-mail και γενικά κάθε επικοινωνία μέσω e-mail θα αγνοείται.

Η οριστική ημερομηνία παράδοσης του project θα είναι ίδια για όλους χωρίς εξαιρέσεις και θα εμφανισθεί στο eclass (Εργασίες) και ορίζεται η ημέρα εξέτασης του μαθήματος στην εξεταστική του χειμερινού εξαμήνου 2025-2026 στις 23:59. **Τα παραδοτέα δεν μπορούν να αλλαχθούν μετά την καταληκτική ημερομηνία παράδοσης μέσω του eclass.**

Μετά την παρέλευση της παράδοσης του project, κανένα project δε θα γίνεται δεκτό μέσω e-mail, με οποιαδήποτε δικαιολογία. Εάν θέλετε να ελέγξετε πως δουλεύει η παράδοση ή εάν δουλεύει ορθά μέσω του eclass, μπορείτε καθόλη τη διάρκεια του χρονικού διαστήματος από την εκφώνηση του 2ου project μέχρι την εξέταση του μαθήματος, να υποβάλλεται όσες φορές επιθυμείτε εργασίες, αλλά θα προσμετρηθεί μόνο η τελευταία (οπότε δε θα υπάρχει δικαιολογία ότι κάτι δεν έγινε καλά εάν δραστηριοποιηθείτε έγκαιρα).

Περιπτώσεις αντιγραφής και λογοκλοπής

Projects που είναι προϊόντα αντιγραφής ή λογοκλοπής θα μηδενίζονται και ανάλογα με τη σοβαρότητα της περίπτωσης το θέμα θα παραπέμπεται στη Επιτροπή Δεοντολογίας του Τμήματος.

Όπως αναφέρεται και στις διαφάνειες της εισαγωγικής διάλεξης για τα διαδικαστικά του μαθήματος: «Η αντιγραφή α-πα-γο-ρεύ-ε-ται!! Τι σημαίνει «αντιγραφή»; Αντιγραφή από έναν συμφοιτητή σας, αντιγραφή από web sites (π.χ., ChatGPT, Stack Overlflow), βοήθεια από «φροντιστήρια». Θα ελέγχουμε με αυτόματο τρόπο για αντιγραφή. Το να μετονομάσετε μεταβλητές ή συναρτήσεις, δεν ξεγελάει τον έλεγχο! Σε περίπτωση αντιγραφής μηδενίζεστε στην εν λόγω άσκηση ΚΑΙ αφαιρείται μία ακόμη μονάδα από τον τελικό βαθμό. Δηλαδή, η αντιγραφή σε ΜΙΑ άσκηση στοιχίζει ΔΥΟ ασκήσεις! Αυτό ισχύει για όλους τους εμπλεκόμενους φοιτητές!»

Οτιδήποτε αναφέρετε στο project θα πρέπει να έχει αναφορά στις πηγές σας με τον ορθό τρόπο. Αν κάτι δεν αναφέρεται, θεωρείται αυτόματα ότι είναι προϊόν δική σας πνευματικής εργασίας. Αν όμως εντοπιστεί ότι κάτι τέτοιο δεν ισχύει, τότε το project μηδενίζεται αυτόματα, ασχέτως από την έκταση του φαινομένου (δηλαδή αν αφορά μόνο ένα σχήμα, μια εικόνα, ή ένα ολόκληρο παραδοτέο).

Στην περίπτωση που κάποιο τμήμα του έργου έχει βασιστεί σε κάτι που αναφέρετε, θα πρέπει να διευκρινίζεται λεπτομερώς και με τρόπο που δεν επιδέχεται αμφισβήτηση τι ακριβώς διαφοροποιεί η δική σας δουλειά.

Βαθμολογία

I. Για το τρέχων πρόγραμμα σπουδών η βαθμολογία στο μάθημα θα προέρχεται:

Τελικός Βαθμός Μαθήματος = Βαθμός Γραπτής Εξέτασης * 50% + Project 1 * 25% + Project 2 * 25%

ή

ΤΒ = ΒΓ*50% + Pr1*25% + Pr2*25%,

Όπου υποχρεωτικά μόνο ο ΒΓ>=5

Επομένως, εάν κάποιος ΔΕΝ παραδώσει ασκήσεις, τότε μπορεί να δώσει γραπτή εξέταση διεκδικώντας τον ανάλογο βαθμό σύμφωνα με το παραπάνω τυπολόγιο υπολογισμού.

Υπενθυμίζεται ότι σε ότι αφορά τη γραπτή εξέταση στο τέλος του εξαμήνου, το 4.75 στρογγυλοποιείται στο 5. Το 4.5, όμως, όχι! Επίσης, το μάθημα είναι αδιαίρετο (όχι «à la carte») κι εάν δεν το περάσετε στο τρέχον ακαδημαϊκό έτος, δίνετε τα project εκ νέου σε επόμενο ακαδημαϊκό έτος.

II. Για το παλιό πρόγραμμα σπουδών

- Λειτουργικά συστήματα Ι

Τελικός Βαθμός Μαθήματος = Βαθμός Γραπτής Εξέτασης * 75% + Project X * 25%, όπου X=1 ή 2

ή

ΤΒ = ΒΓ*75% + PrX*25%, , όπου X=1 ή 2

Αυτό σημαίνει ότι υποχρεωτικό είναι μόνο ένα από τα δύο projects κι εφόσον κάποιος παραδώσει και τα δύο θα μετρήσει ο καλύτερος βαθμός.

- Εργαστήριο Λειτουργικών Συστημάτων

Τελικός Βαθμός Μαθήματος = Project 1 * 50% + Project 2 * 50%

ή

ΤΒ = Pr1*50% + Pr2*50%

Που σημαίνει υποχρέωση παράδοσης και των δύο projects και δε θα υπάρχει τελική εξέταση, αλλά ο μέσος όρος των ασκήσεων είναι ο τελικός βαθμός στο εργαστήριο.

Εργαστηριακή Άσκηση

Επέκταση Χρονοπρογραμματιστή Διεργασιών

1. Εισαγωγή

1.1 Αντικείμενο Εργασίας

Το ζητούμενο στην εργασία είναι η επέκταση ενός περιβάλλοντος χρονοπρογραμματισμού σε λειτουργικό σύστημα Unix. Συγκεκριμένα, πρέπει να επεκταθεί ένας δρομολογητής ο οποίος παίρνει σαν είσοδο τις εφαρμογές που πρέπει να εκτελέσει, διαβάζοντας ένα αρχείο με τα ονόματα και τις ιδιότητές τους (π.χ. προτεραιότητα), τις εισάγει σε μία κατάλληλη δομή δεδομένων (λίστα) και τις δρομολογεί ανάλογα με την πολιτική δρομολόγησης, η οποία μπορεί να επιτρέπει ή όχι διακοπή της τρέχουσας διεργασίας. Στη δεύτερη φάση, ο δρομολογητής θα πρέπει να χειρίζεται την περίπτωση που μια διεργασία εκτελεί I/O.

1.2 Δεδομένα Εργασίας

Σας δίνεται η υλοποίηση ενός δρομολογητή για τις δύο βασικές πολιτικές δρομολόγησης (FCFS, RR) και την περίπτωση που υπάρχει ένας μόνο επεξεργαστής στο σύστημα.

Η υλοποίηση του δρομολογητή έχει 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

1.2.1 Περιγραφέας εφαρμογής

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

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

- **NEW**: 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.

1.2.2 Ουρά Εκτέλεσης

Η δομή δεδομένων που περιέχει τους περιγραφείς των εφαρμογών είναι μία συνδεδεμένη λίστα (ουρά). Εισαγωγή διεργασιών πραγματοποιείται στο τέλος της λίστας ενώ εξαγωγή από την αρχή της.

1.2.3 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 στατικής πολιτικής δρομολόγησης (FCFS), ο δρομολογητής ενεργοποιείται μόνο όταν μια εφαρμογή – διεργασία τερματίζει.

Αντίθετα, στην περίπτωση δυναμικής πολιτικής δρομολόγησης (RR), ο δρομολογητής ενεργοποιείται κατά τακτά χρονικά διαστήματα, όπως καθορίζεται από το κβάντο δρομολόγησης που αποτελεί παράμετρο του προγράμματος. Ο μηχανισμός ενεργοποίησης του δρομολογητή υλοποιείται 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, πραγματοποιείται με χρήση των σημάτων **SIGSTOP** και **SIGCONT**.

Τέλος, όταν μια εφαρμογή – διεργασία τερματίζει, ο δρομολογητής 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 τότε υπολογίζεται και εκτυπώνεται ο συνολικός χρόνος εκτέλεσής της, ξεκινώντας από τη στιγμή που μπήκε στην λίστα εκτέλεσης μέχρι τη στιγμή που ολοκληρώθηκε η εκτέλεσή της. Στην αρχική έκδοση του κώδικα, όλες οι διεργασίες εισάγονται στο σύστημα, δηλαδή στην ουρά εκτέλεσης, πριν ξεκινήσει η εκτέλεση της πολιτικής δρομολόγησης (δηλαδή την αντίστοιχη χρονική στιγμή "0" σύμφωνα με τις ασκήσεις θεωρίας).

1.2.4 Συμπληρωματικές Πληροφορίες

1. Ο δρομολογητής λαμβάνει το σήμα **SIGCHLD** μόνο όταν μια εφαρμογή – παιδί τερματίζει και όχι όταν αναστέλλεται η εκτέλεσή της εφαρμογής - παιδιού (βλ. **sigaction**)
2. Ο δρομολογητής διατηρεί σε κάποια μεταβλητή έναν δείκτη στον περιγραφέα της διεργασίας που εκτελείται μια δεδομένη χρονική στιγμή.
3. Ο δρομολογητής εμφανίζει πληροφορίες για την εκτέλεση κάθε εφαρμογής που έχει ολοκληρωθεί. Ο δρομολογητής εμφανίζει και το συνολικό χρόνο εκτέλεσης του συνόλου των εφαρμογών.
4. Ο κώδικας των παραπάνω δοκιμαστικών προγραμμάτων, χαρακτηριστικά αρχεία εισόδου, ένα αρχείο (script) με παραδείγματα εκτέλεσης του δρομολογητή, ένα ενδεικτικό παράδειγμα εκτύπωσης αποτελεσμάτων, καθώς και ο αρχικός κώδικας του δρομολογητή, τον οποίο πρέπει να επεκτείνετε, δίνονται μαζί με την εκφώνηση της άσκησης.

1.2.5. Χρήση

Σύμφωνα με τα παραπάνω, το πρόγραμμα του δρομολογητή 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, RR.
- **quantum:** Το κβάντο δρομολόγησης σε msec. Απαιτείται μόνο για την πολιτική δρομολόγησης RR.
- **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.

Παραδείγματα χρήσης:
$ scheduler FCFS homogeneous.txt
$ scheduler FCFS reverse.txt
$ scheduler RR 1000 homogeneous.txt
$ scheduler RR 1000 reverse.txt

2. Πρώτη Φάση: υποστήριξη προτεραιοτήτων και διαφορετικού χρόνου έναρξης

Ο αρχικός δρομολογητής θεωρεί πως η εισαγωγή των εργασιών στην ουρά εκτέλεσης γίνεται στη χρονική στιγμή μηδέν και όλες οι εργασίες έχουν την ίδια προτεραιότητα. Στην πρώτη φάση της εργασίας, θα πρέπει να επεκτείνετε τον αρχικό δρομολογητή ώστε να διαχειρίζεται διεργασίες με διαφορετικές προτεραιότητες, ενώ οι εφαρμογές μπορούν να εισέρχονται στο σύστημα σε διαφορετικές χρονικές στιγμές. Η προτεραιότητα κάθε εφαρμογής θα δίνεται στο αρχείο εισόδου που περιγράφει το φορτίο εργασιών, δίπλα στο όνομα του εκτελέσιμου μιας εφαρμογής και με τιμή από 0 (χαμηλότερη προτεραιότητα) και 10 (υψηλότερη προτεραιότητα). Αν δεν έχει οριστεί η προτεραιότητα, τότε η εφαρμογή θα έχει προτεραιότητα 0, αλλά για απλότητα μπορείτε να θεωρήσετε πως ο ορισμός προτεραιότητας απαιτείται πάντα. Το αρχείο εισόδου θα έχει και εντολές της μορφής "*sleep X*", που σημαίνουν πως το τμήμα του δρομολογητή που διαβάζει και εισάγει τις διεργασίες στο σύστημα θα πρέπει να περιμένει για Χ δευτερόλεπτα προτού επεξεργαστεί την επόμενη γραμμή του αρχείου.

Σύμφωνα με τα παραπάνω, ο δρομολογητής παίρνει σαν είσοδο το αρχείο με τις εφαρμογές που πρέπει να εκτελέσει, διαβάζοντας σε κάθε γραμμή είτε το όνομα και την προτεραιότητα της διεργασίας που θα εισάγει σε μία κατάλληλη δομή δεδομένων (λίστα) είτε εκτελώντας κάποια εντολή αναμονής (sleep). Ταυτόχρονα θα δρομολογεί τις ενεργές εργασίας εφαρμόζοντας μία από τις ακόλουθες πολιτικές:

- **Εξυπηρέτηση με βάση τη σειρά άφιξης (FCFS_PNP) και την προτεραιότητα – χωρίς προεκχώρηση**: οι εφαρμογές εκτελούνται με τη σειρά εισαγωγής τους στη λίστα, δηλαδή με τη σειρά που αναγράφονται στο αρχείο εισόδου. Η εκτέλεση μιας διεργασίας πραγματοποιείται χωρίς διακοπή μέχρι την ολοκλήρωσή της, ακόμα και αν κατά την εκτέλεσή της εισαχθεί στο σύστημα διεργασία με μεγαλύτερη προτεραιότητα. Μετά την ολοκλήρωση μιας διεργασίας, ο δρομολογητής επιλέγει από της ουρά την διεργασία με την μεγαλύτερη προτεραιότητα, διαφορετικά τη διεργασία που είχε εισαχθεί νωρίτερα στην ουρά.

- **Εξυπηρέτηση με βάση τη σειρά άφιξης (FCFS_PRIO) και την προτεραιότητα – με προεκχώρηση**: οι εφαρμογές εκτελούνται με τη σειρά εισαγωγής τους στη λίστα, δηλαδή με τη σειρά που αναγράφονται στο αρχείο εισόδου. Η εκτέλεση μιας διεργασίας μπορεί να διακοπεί πριν την ολοκλήρωσή της αν κατά την εκτέλεσή της εισαχθεί στο σύστημα διεργασία με μεγαλύτερη προτεραιότητα. Μετά την ολοκλήρωση μιας διεργασίας, ο δρομολογητής επιλέγει από της ουρά την διεργασία με την μεγαλύτερη προτεραιότητα, διαφορετικά τη διεργασία που είχε εισαχθεί νωρίτερα στην ουρά.

Για τη συγκεκριμένη επέκταση, ο κώδικας του δρομολογητή θα υλοποιηθεί στο αρχείο **scheduler_v1.c.** Η χρήση της ανανεωμένου δρομολογητή παραμένει όπως πριν, σύμφωνα με τα παρακάτω παραδείγματα:
$ scheduler_v2 FCFS_PNP priority.txt
$ scheduler_v2 FCFS_PRIO priority.txt

2. Δεύτερη Φάση: υποστήριξη I/O

Μέχρι στιγμής έχουμε θεωρήσει ότι οι προς δρομολόγηση εφαρμογές δεν περιλαμβάνουν εντολές I/O και κατά συνέπεια δεν παραχωρούν τον επεξεργαστή παρά μόνο όταν τερματίσουν ή λήξει το κβάντο χρόνου κατά το οποίο έχουν επιλεγεί να εκτελεστούν. Στο συγκεκριμένο ερώτημα καλείστε να επεκτείνετε τον **αρχικό δρομολογητή** ώστε να λαμβάνει υπόψη του εφαρμογές που πρόκειται να εκτελέσουν εντολές I/O. Η εξομοίωση της παραπάνω περίπτωσης γίνεται με χρήση κατάλληλων σημάτων. Συγκεκριμένα, μια εφαρμογή, που προφανώς βρίσκεται σε κατάσταση εκτέλεσης, καλεί μια ρουτίνα **perform_io(msec)** που πραγματοποιεί τα ακόλουθα:

1. Ενημερώνει τον δρομολογητή ότι πρόκειται να ξεκινήσει I/O αποστέλλοντας του το σήμα **SIGUSR1**
2. «Κοιμάται» (βλ. **sleep ή nanosleep**) για ορισμένο χρονικό διάστημα, όπως ορίζεται από το όρισμα της ρουτίνας.
3. Αποστέλλει στο δρομολογητή ένα δεύτερο σήμα (**SIGUSR2**) ενημερώνοντας τον ότι έχει ολοκληρώσει το I/O και μπορεί να συνεχίσει την εκτέλεση της.
4. Αποστέλλει στον εαυτό της το σήμα **SIGSTOP** ώστε να ενεργοποιηθεί πάλι από τον δρομολογητή, όταν αυτός την επιλέξει για εκτέλεση.

Είναι προφανές ότι ο κώδικας του αρχικού δρομολογητή πρέπει να επεκταθεί κατάλληλα ώστε σε κάθε περίπτωση να εκτελεί τις κατάλληλες ενέργειες (π.χ. δρομολόγηση μιας άλλης εφαρμογής, εισαγωγή στην ουρά εκτέλεσης μιας εφαρμογής που έχει ολοκληρώσει τις εντολές I/O, κλπ). Η λίστα καταστάσεων μίας διεργασίας μπορεί επίσης να επεκταθεί.

Για τη συγκεκριμένη επέκταση, ο κώδικας του δρομολογητή θα υλοποιηθεί στο αρχείο **scheduler_v2.c.** Το δοκιμαστικό πρόγραμμα **work5x2_io**, που δημιουργείται με μεταγλώττιση του κώδικα **work_io.c** χρησιμοποιεί τον μηχανισμό προσομοίωσης I/O που παρουσιάστηκε παραπάνω. Επίσης το αρχείο εκτέλεσης **mixed.txt** περιλαμβάνει ένα workload που μπορεί να χρησιμοποιηθεί για τη αξιολόγηση της ορθότητας της υλοποίησή σας.

Παρατήρηση: το συγκεκριμένο ερώτημα αρκεί να υλοποιηθεί στην νέα έκδοση (v2) του δρομολογητή μόνο για την πολιτική FCFS. Αν το υλοποιήσετε σωστά και τις δύο πολιτικές (FCFS) τότε θα δοθεί μία έξτρα μονάδα στη βαθμολόγηση της εργασίας.

Η έκδοση v2 του δρομολογητή είναι εντελώς ανεξάρτητη από την έκδοση v1 της πρώτης φάσης**.** Η χρήση του ανανεωμένου δρομολογητή παραμένει όπως στην αρχική έκδοση (v0), σύμφωνα με τα παρακάτω παραδείγματα:
$ scheduler_v2 FCFS mixed.txt
$ scheduler_v2 RR 1000 mixed.t

4. Σύνοψη και παράδοση εργασίας

4.1 Ζητούμενα

Καλείστε να συμπληρώστε τα ζητούμενα προγράμματα δρομολόγησης, τα εκτελέσιμα των οποίων θα ονομάσετε **scheduler_v1** (πρώτη έκδοση) και **scheduler_v2** (δεύτερη έκδοση).

Αναλυτικότερα:

1. Μελετήστε προσεκτικά τον αρχικό κώδικα του χρονοδρομολογητή.
2. Επεκτείνετε τις δομές και λειτουργίες του βασικού χρονοδρομολογητή για κάθε περίπτωση.
3. Εκτελέστε και αξιολογήστε τις νέες πολιτικές δημιουργώντας παραδείγματα εκτέλεσης του δρομολογητή αντίστοιχα με αυτά που υπάρχει αρχικά στο **run.sh**.
4. Υλοποιήστε την δεύτερη φάση του δρομολογητή.
5. 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:

- Συμπεριλάβετε στο αρχείο **zip** (**όχι rar**) με τον κώδικά σας, τη δομή (κατάλογοι και αρχεία) του αρχικού κώδικα που σας έχει δοθεί. **Πρέπει να δηλαδή να περιέχει τους δύο καταλόγους, τα αντίστοιχα αρχεία Makefile, τον κώδικα των βοηθητικών προγραμμάτων και τον συμπληρωμένο κώδικα των δύο υλοποιήσεων του δρομολογητή**.
- **Μην συμπεριλάβετε εκτελέσιμα αρχεία** στο αρχείο **zip** με τον κώδικα σας.
- Μπορείτε να κάνετε οποιεσδήποτε αλλαγές στον αρχικό κώδικα του δρομολογητή αλλά θα πρέπει να ακολουθήσετε την 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 όπως 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 δοκιμαστεί σε συστήματα Linux (βλ. falcon) χωρίς προβλήματα.
- Αν εκτελέσετε τα πειράματά σας σε σύστημα το οποίο χρησιμοποιείται από ταυτόχρονα από πολλούς χρήστες (π.χ. falcon), τότε ο χρόνος εκτέλεσης των βοηθητικών προγραμμάτων αναμένεται να διαφέρει κάθε φορά, ανάλογα με το συνολικό φόρτο του συστήματος.

Καλή επιτυχία!