ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
http://www.cslab.ece.ntua.gr

Λειτουργικά Συστήματα

7ο εξάμηνο, Ακαδημαϊκή περίοδος 2016-2017

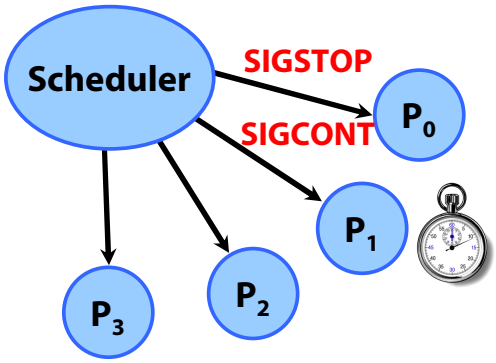
Άσκηση 4: Χρονοδρομολόγηση

1 Ασκήσεις

1.1 Υλοποίηση χρονοδρομολογητή κυκλικής επαναφοράς στο χώρο χρήστη

Ζητείται η υλοποίηση ενός χρονοδρομολογητή κυκλικής επαναφοράς (round-robin). Ο χρονοδρομολογητής εκτελείται ως γονική διεργασία, στο χώρο χρήστη, κατανέμοντας τον υπολογιστικό χρόνο σε διεργασίες-παιδιά. Για τον έλεγχο των διεργασιών ο χρονοδρομολογητής θα χρησιμοποιεί τα σήματα `SIGSTOP` και `SIGCONT` για τη διακοπή και την ενεργοποίηση κάθε διεργασίας, αντίστοιχα.

Κάθε διεργασία εκτελείται για χρονικό διάστημα το πολύ ίσο με το *κβάντο χρόνου* `tq`. Αν η διεργασία τερματιστεί πριν από το τέλος του κβάντου χρόνου, ο χρονοδρομολογητής την αφαιρεί από την ουρά των έτοιμων διεργασιών και ενεργοποιεί την επόμενη. Αν το κβάντο χρόνου εκπνεύσει χωρίς η διεργασία να έχει ολοκληρώσει την εκτέλεσή της, τότε αυτή διακόπτεται, τοποθετείται στο τέλος της ουράς έτοιμων διεργασιών και ενεργοποιείται η επόμενη.



Σχήμα 1: Ο χρονοδρομολογητής σταματά την P_0 και ξεκινά την P_1.

Η λειτουργία του χρονοδρομολογητή ζητείται να είναι *ασύγχρονη*, βασισμένη σε σήματα. Ένας χρονοδρομολογητής χώρου πυρήνα ενεργοποιείται από διακοπές χρονιστή. Αντίστοιχα, ο υπό εξέταση χρονοδρομολογητής θα χρησιμοποιεί τα σήματα `SIGALRM` και `SIGCHLD` για να ενεργοποιείται στις εξής δύο περιπτώσεις:

- **Εκπνοή κβάντου χρόνου:** Όταν το κβάντο χρόνου εκπνεύσει, ο χρονοδρομολογητής σταματά την τρέχουσα διεργασία. Η περίπτωση αυτή αντιστοιχεί σε χειρισμό του σήματος `SIGALRM`.
- **Παύση/τερματισμός διεργασίας:** Όταν η τρέχουσα διεργασία πεθάνει ή σταματήσει, επειδή εξέπνευσε το κβάντο χρόνου της, ο χρονοδρομολογητής διαλέγει την επόμενη από την ουρά, θέτει τον χρονιστή ώστε να παραδοθεί σήμα `SIGALRM` μετά από `tq` δευτερόλεπτα, και την ενεργοποιεί. Η περίπτωση αυτή αντιστοιχεί σε χειρισμό του σήματος `SIGCHLD`.

Το τελικό παραδοτέο είναι το πρόγραμμα του χρονοδρομολογητή, το οποίο θα εκτελείται με ορίσματα τα εκτελέσιμα προς χρονοδρομολόγηση:

```
$ ./scheduler prog prog prog prog
```

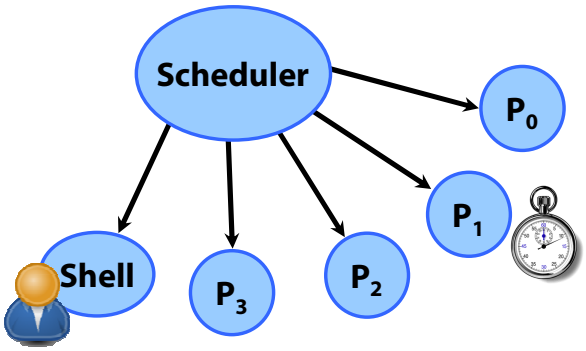
Για κάθε όρισμα κατασκευάζεται μία διεργασία που εκτελεί το αντίστοιχο πρόγραμμα. Σε κάθε διεργασία αντιστοιχίζεται σειριακός αριθμός `id`. Μετά τη δημιουργία των διεργασιών ξεκινά η διαδικασία χρονοδρομολόγησης. Ο χρονοδρομολογητής εμφανίζει κατάλληλα μηνύματα κατά την ενεργοποίηση, διακοπή και τερματισμό των διεργασιών. Όταν όλες οι διεργασίες έχουν ολοκληρώσει την εκτέλεσή τους, τερματίζεται.

Στον κατάλογο `/home/oslab/code/sched` σας δίνονται: το πρόγραμμα `prog.c`, το οποίο θα εκτελείται στις υπό χρονοδρομολόγηση διεργασίες, το `execve-example.c` για τη χρήση της κλήσης συστήματος `execve()` και ένας σκελετός του χρονοδρομολογητή, `scheduler.c`.

Σημείωση: Μια διεργασία που εκτελείται υπό την `strace` *αγνοεί* το σήμα `SIGSTOP`. Οπότε, αποφύγετε τη χρήση της `strace` πάνω σε παιδιά, αφού η χρονοδρομολόγηση γίνεται με `SIGSTOP` και `SIGCONT`. Αντίθετα, μπορείτε να εκτελέσετε τον χρονοδρομολογητή υπό την `strace` χωρίς κανένα πρόβλημα.

1.2 Έλεγχος λειτουργίας χρονοδρομολογητή μέσω φλοιού

Ζητείται η επέκταση του χρονοδρομολογητή του προηγούμενου ερωτήματος, ώστε να υποστηρίζεται ο έλεγχος της λειτουργίας του μέσω προγράμματος-φλοιού. Ο χρήστης



Σχήμα 2: Το πρόγραμμα-φλοιός ελέγχει τη λειτουργία του χρονοδρομολογητή

του συστήματος έχει τη δυνατότητα να ζητά δυναμική δημιουργία και τερματισμό διεργασιών, αλληλεπιδρώντας με το πρόγραμμα του φλοιού.

Ο φλοιός δέχεται εντολές από το χρήστη, κατασκευάζει αιτήσεις κατάλληλης μορφής τις οποίες αποστέλλει προς τον χρονοδρομολογητή, λαμβάνει απαντήσεις που ενημερώνουν για την έκβαση της εκτέλεσής τους (επιτυχία / αποτυχία) και ενημερώνει για το αποτέλεσμά τους το χρήστη.

Ο φλοιός διαθέτει τέσσερις εντολές:[1]

- **Εντολή 'p':** Ο χρονοδρομολογητής εκτυπώνει στην έξοδο λίστα με τις υπό εκτέλεση διεργασίες, στον οποίο φαίνεται ο σειριακός αριθμός `id` της διεργασίας, το `PID` και το όνομα της. Επιπλέον, επισημαίνεται η τρέχουσα διεργασία.
- **Εντολή 'k':** Δέχεται όρισμα το `id` μιας διεργασίας (*προσοχή:* όχι το `PID`) και ζητά από το χρονοδρομολογητή τον τερματισμό της.
- **Εντολή 'e':** Δέχεται όρισμα το όνομα ενός εκτελέσιμου στον τρέχοντα κατάλογο, π.χ. `prog2` και ζητά τη δημιουργία μιας νέας διεργασίας από τον χρονοδρομολογητή, στην οποία θα τρέχει αυτό το εκτελέσιμο.
- **Εντολή 'q':** Ο φλοιός τερματίζει τη λειτουργία του.

Τελικό παραδοτέο είναι νέα έκδοση του χρονοδρομολογητή, η οποία θα εξυπηρετεί τις αιτήσεις του φλοιού. Ο φλοιός ζητείται να χρονοδρομολογείται *μαζί* με τις υπόλοιπες διεργασίες, όντας στην ουρά εκτέλεσης και λαμβάνοντας κβάντα χρόνου σύμφωνα με τον αλγόριθμο RR. Ο χρονοδρομολογητής θα τερματίζεται όταν όλες οι διεργασίες που χειρίζεται τερματίσουν, συμπεριλαμβανομένου του φλοιού.

Σας δίνεται το πρόγραμμα του φλοιού, `shell.c` και το αρχείο `request.h` που ορίζει τη μορφή των αιτήσεων από το φλοιό προς τον χρονοδρομολογητή με βάση τη δομή `struct request_struct`. Για την υποστήριξη του φλοιού είναι απαραίτητος ένας μηχανισμός επικοινωνίας από το φλοιό προς τον χρονοδρομολογητή και αντίστροφα. Ο μηχανισμός αυτός δίνεται στο αρχείο `scheduler-shell.c`, που αποτελεί σκελετό της ζητούμενης υλοποιήσης.

1.3 Υλοποίηση προτεραιοτήτων στο χρονοδρομολογητή

Ζητείται η επέκταση του χρονοδρομολογητή του προηγούμενου ερωτήματος, ώστε να υποστηρίζονται δύο κλάσεις προτεραιότητας: *HIGH* και *LOW*. Ο αλγόριθμος χρονοδρομολόγησης αλλάζει ως εξής: αν υπάρχουν διεργασίες προτεραιότητας *HIGH* εκτελούνται

[1] Υπάρχουν ακόμα δύο εντολές (**h** και **l**), οι οποίες όμως αφορούν το επόμενο ερώτημα και μπορούν, προς το παρόν, να αγνοηθούν.

μόνο αυτές χρησιμοποιώντας κυκλική επαναφορά (round-robin). Σε αντίθετη περίπτωση, χρονοδρομολογούνται οι *LOW* διεργασίες χρησιμοποιώντας κυκλική επαναφορά. Όλες οι διεργασίες δημιουργούνται με *LOW* προτεραιότητα. Η αλλαγή της προτεραιότητας μιας διεργασίες θα πραγματοποιείται με τις παρακάτω εντολές φλοιού:

- **'h'** (**'l'**) Δέχεται όρισμα το `id` μιας διεργασίας και θέτει την προτεραιότητα της σε *HIGH* (*LOW*).

2 Εξέταση άσκησης και αναφορά

Η εξέταση της άσκησης θα πραγματοποιηθεί σε ημερομηνία που θα ανακοινωθεί στη λίστα του μαθήματος.

Μετά την εξέταση η κάθε ομάδα θα πρέπει να συντάξει μια (σύντομη) αναφορά και να τη στείλει μέσω e-mail στους υπευθύνους των εργαστηριακών ομάδων. Η προθεσμία για την αναφορά είναι μια εβδομάδα μετά την προθεσμία εξέτασης της άσκησης.

Η αναφορά αυτή θα περιέχει:

- Τον πηγαίο κώδικα (source code) των ασκήσεων.
- ενδεικτική έξοδο εκτέλεσης των προγραμμάτων.
- Σύντομες απαντήσεις στις ερωτήσεις.

3 Ερωτήσεις Αναφοράς

3.1 Άσκηση 1.1

1. Τι συμβαίνει αν το σήμα `SIGALRM` έρθει ενώ εκτελείται η συνάρτηση χειρισμού του σήματος `SIGCHLD` ή το αντίστροφο; Πώς αντιμετωπίζει ένας πραγματικός χρονοδρομολογητής χώρου πυρήνα ανάλογα ενδεχόμενα και πώς η δική σας υλοποίηση; *Υπόδειξη:* μελετήστε τη συνάρτηση `install_signal_handlers()` που δίνεται.
2. Κάθε φορά που ο χρονοδρομολογητής λαμβάνει σήμα `SIGCHLD`, σε ποια διεργασία-παιδί περιμένετε να αναφέρεται αυτό; Τι συμβαίνει αν λόγω εξωτερικού παράγοντα (π.χ. αποστολή `SIGKILL`) τερματιστεί αναπάντεχα μια οποιαδήποτε διεργασία-παιδί;
3. Γιατί χρειάζεται ο χειρισμός *δύο* σημάτων για την υλοποίηση του χρονοδρομολογητή; θα μπορούσε ο χρονοδρομολογητής να χρησιμοποιεί μόνο το σήμα `SIGALRM` για να σταματά την τρέχουσα διεργασία και να ξεκινά την επόμενη; Τι ανεπιθύμητη συμπεριφορά θα μπορούσε να εμφανίζει μια τέτοια υλοποίηση; *Υπόδειξη:* Η παραλαβή του σήματος `SIGCHLD` εγγυάται ότι η τρέχουσα διεργασία έλαβε το σήμα `SIGSTOP` και έχει σταματήσει.

3.2 Άσκηση 1.2

1. Όταν και ο φλοιός υφίσταται χρονοδρομολόγηση, ποια εμφανίζεται πάντοτε ως τρέχουσα διεργασία στη λίστα διεργασιών (**εντολή 'p'**); Θα μπορούσε να μη συμβαίνει αυτό; Γιατί;
2. Γιατί είναι αναγκαίο να συμπεριλάβετε κλήσεις `signals_disable()`, `_enable()` γύρω από την συνάρτηση υλοποίησης αιτήσεων του φλοιού; *Υπόδειξη:* Η συνάρτηση υλοποίησης αιτήσεων του φλοιού μεταβάλλει δομές όπως η ουρά εκτέλεσης των διεργασιών.

3.3 Άσκηση 1.3

1. Περιγράψτε ένα σενάριο δημιουργίας λιμοκτονίας.

4 Προαιρετικές ερωτήσεις (επιπλέον βαθμοί: 0)

1. Περιγράψτε τον μηχανισμό επικοινωνίας μεταξύ του φλοιού και του χρονοδρομολογητή στην Άσκηση 1.2.

2. Προδιαγράψτε και σκιαγραφήστε υλοποίηση μηχανισμού γήρανσης για την αποφυγή λιμοκτονίας στην Άσκηση 1.3.

3. Έστω χρονοδρομολογητής που χρησιμοποιεί πολιτική χρονοδρομολόγησης αντίστοιχη με αυτή της Άσκησης 1.3, και υποστηρίζει τρεις κλάσεις προτεραιότητας: *HIGH*, *MEDIUM* και *LOW*. Έστω, επίσης, ότι ο χρονοδρομολογητής αναλαμβάνει να αναστέλλει την εκτέλεση διεργασιών, όταν αυτές πρέπει να περιμένουν σε κάποιο σημαφόρο. Έστω τρεις διεργασίες `H`, `M`, `L` με προτεραιότητες *HIGH*, *MEDIUM*, *LOW*, αντίστοιχα.

   Αναγνωρίστε το πρόβλημα που προκύπτει στην παρακάτω αλληλουχία γεγονότων:

   i. Ο σημαφόρος `S` αρχικοποιείται στην τιμή 1
   ii. Αρχικά εκτελείται μόνο η `L`, η οποία εκτελεί `S.wait()`
   iii. Πριν η `L` απελευθερώσει τον σημαφόρο, αρχίζει να εκτελείται η `M`, η οποία εκτελεί αέναο βρόχο (infinite loop)
   iv. Αρχίζει να εκτελείται η `H`, η οποία εκτελεί `S.wait()`

   Προτείνετε τρόπους αντιμετώπισης του προβλήματος.