4η Εργαστηριακή Άσκηση:
Χρονοδρομολόγηση

1 Ασκήσεις

1.1 Υλοποίηση χρονοδρομολογητή κυκλικής επαναφοράς στο χώρο χρήστη

```
george@georgepag4028:~/Desktop/code/os/ask4/p4.1$ ./scheduler prog prog
My PID = 6390: Child PID = 6391 has been stopped by a signal, signo = 19
My PID = 6390: Child PID = 6392 has been stopped by a signal, signo = 19
prog: Starting, NMSG = 20, delay = 145
prog[6391]: This is message 0
prog[6391]: This is message 1
prog[6391]: This is message 2
prog[6391]: This is message 3
prog[6391]: This is message 4
prog[6391]: This is message 5
prog[6391]: This is message 6
My PID = 6390: Child PID = 6391 has been stopped by a signal, signo = 19
prog: Starting, NMSG = 20, delay = 126
prog[6392]: This is message 0
prog[6392]: This is message 1
prog[6392]: This is message 2
prog[6392]: This is message 3
prog[6392]: This is message 4
prog[6392]: This is message 5
prog[6392]: This is message 6
prog[6392]: This is message 7
My PID = 6390: Child PID = 6392 has been stopped by a signal, signo = 19
prog[6391]: This is message 7
prog[6391]: This is message 8
prog[6391]: This is message 9
prog[6391]: This is message 10
prog[6391]: This is message 11
prog[6391]: This is message 12
My PID = 6390: Child PID = 6391 has been stopped by a signal, signo = 19
prog[6392]: This is message 8
prog[6392]: This is message 9
prog[6392]: This is message 10
prog[6392]: This is message 11
prog[6392]: This is message 12
prog[6392]: This is message 13
prog[6392]: This is message 14
My PID = 6390: Child PID = 6392 has been stopped by a signal, signo = 19
prog[6391]: This is message 13
prog[6391]: This is message 14
prog[6391]: This is message 15
prog[6391]: This is message 16
prog[6391]: This is message 17
prog[6391]: This is message 18
My PID = 6390: Child PID = 6391 has been stopped by a signal, signo = 19
prog[6392]: This is message 15
prog[6392]: This is message 16
prog[6392]: This is message 17
prog[6392]: This is message 18
prog[6392]: This is message 19
My PID = 6390: Child PID = 6392 terminated normally, exit status = 0
prog[6391]: This is message 19
My PID = 6390: Child PID = 6391 terminated normally, exit status = 0
Done!
george@georgepag4028:~/Desktop/code/os/ask4/p4.1$
```

```
george@georgepag4028:~/Desktop/code/os/ask4/p4.1$ ./prog
./prog: Starting, NMSG = 20, delay = 46
./prog[6467]: This is message 0
./prog[6467]: This is message 1
./prog[6467]: This is message 2
./prog[6467]: This is message 3
./prog[6467]: This is message 4
./prog[6467]: This is message 5
./prog[6467]: This is message 6
./prog[6467]: This is message 7
./prog[6467]: This is message 8
./prog[6467]: This is message 9
./prog[6467]: This is message 10
./prog[6467]: This is message 11
./prog[6467]: This is message 12
./prog[6467]: This is message 13
./prog[6467]: This is message 14
./prog[6467]: This is message 15
./prog[6467]: This is message 16
./prog[6467]: This is message 17
./prog[6467]: This is message 18
./prog[6467]: This is message 19
george@georgepag4028:~/Desktop/code/os/ask4/p4.1$
```

1.2 Έλεγχος λειτουργίας χρονοδρομολογητή μέσω φλοιού

```
george@georgepag4028:~/Desktop/code/os/ask4/p4.2$ ./scheduler-shell prog
My PID = 6521: Child PID = 6522 has been stopped by a signal, signo = 19
My PID = 6521: Child PID = 6523 has been stopped by a signal, signo = 19


scheduler-shell(6521)─┬─prog(6523)
                      ├─sh(6524)───pstree(6525)
                      └─shell(6522)



This is the Shell. Welcome.

Shell> My PID = 6521: Child PID = 6522 has been stopped by a signal, signo = 19
prog: Starting, NMSG = 20, delay = 55
prog[6523]: This is message 0
prog[6523]: This is message 1
prog[6523]: This is message 2
prog[6523]: This is message 3
prog[6523]: This is message 4
prog[6523]: This is message 5
prog[6523]: This is message 6
prog[6523]: This is message 7
prog[6523]: This is message 8
prog[6523]: This is message 9
prog[6523]: This is message 10
prog[6523]: This is message 11
prog[6523]: This is message 12
prog[6523]: This is message 13
prog[6523]: This is message 14
prog[6523]: This is message 15
prog[6523]: This is message 16
prog[6523]: This is message 17
My PID = 6521: Child PID = 6523 has been stopped by a signal, signo = 19
My PID = 6521: Child PID = 6522 has been stopped by a signal, signo = 19
prog[6523]: This is message 18
prog[6523]: This is message 19
My PID = 6521: Child PID = 6523 terminated normally, exit status = 0


scheduler-shell(6521)─┬─sh(6542)───pstree(6543)
                      └─shell(6522)


Done!
george@georgepag4028:~/Desktop/code/os/ask4/p4.2$
```

```
george@georgepag4028:~/Desktop/code/os/ask4/p4.2$ ./scheduler-shell prog prog
My PID = 6591: Child PID = 6592 has been stopped by a signal, signo = 19
My PID = 6591: Child PID = 6593 has been stopped by a signal, signo = 19
My PID = 6591: Child PID = 6594 has been stopped by a signal, signo = 19


scheduler-shell(6591)─┬─prog(6593)
                      ├─prog(6594)
                      ├─sh(6595)───pstree(6596)
                      └─shell(6592)



This is the Shell. Welcome.

Shell> My PID = 6591: Child PID = 6592 has been stopped by a signal, signo = 19
prog: Starting, NMSG = 20, delay = 130
prog[6593]: This is message 0
prog[6593]: This is message 1
prog[6593]: This is message 2
prog[6593]: This is message 3
prog[6593]: This is message 4
prog[6593]: This is message 5
prog[6593]: This is message 6
prog[6593]: This is message 7
My PID = 6591: Child PID = 6593 has been stopped by a signal, signo = 19
prog: Starting, NMSG = 20, delay = 112
prog[6594]: This is message 0
prog[6594]: This is message 1
prog[6594]: This is message 2
prog[6594]: This is message 3
prog[6594]: This is message 4
prog[6594]: This is message 5
prog[6594]: This is message 6
prog[6594]: This is message 7
prog[6594]: This is message 8
My PID = 6591: Child PID = 6594 has been stopped by a signal, signo = 19
My PID = 6591: Child PID = 6592 has been stopped by a signal, signo = 19
prog[6593]: This is message 8
prog[6593]: This is message 9
prog[6593]: This is message 10
prog[6593]: This is message 11
prog[6593]: This is message 12
prog[6593]: This is message 13
prog[6593]: This is message 14
My PID = 6591: Child PID = 6593 has been stopped by a signal, signo = 19
prog[6594]: This is message 9
prog[6594]: This is message 10
prog[6594]: This is message 11
prog[6594]: This is message 12
prog[6594]: This is message 13
prog[6594]: This is message 14
prog[6594]: This is message 15
prog[6594]: This is message 16
My PID = 6591: Child PID = 6594 has been stopped by a signal, signo = 19
My PID = 6591: Child PID = 6592 has been stopped by a signal, signo = 19
prog[6593]: This is message 15
prog[6593]: This is message 16
prog[6593]: This is message 17
prog[6593]: This is message 18
prog[6593]: This is message 19
My PID = 6591: Child PID = 6593 terminated normally, exit status = 0


scheduler-shell(6591)─┬─prog(6594)
                      ├─sh(6609)───pstree(6610)
                      └─shell(6592)


prog[6594]: This is message 17
prog[6594]: This is message 18
prog[6594]: This is message 19
My PID = 6591: Child PID = 6594 terminated normally, exit status = 0


scheduler-shell(6591)─┬─sh(6614)───pstree(6615)
                      └─shell(6592)


Done!
```

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.

Στην περίπτωση του δικού μας χρονοδρομολογητή, αμα έρθουν 2 σήματα ταυτόχρονα, το 2ο ακυρώνεται, γιατί δεν υπάρχει ουρά queue που να κρατάει τα σήματα που έχουν έρθει. Σε αντίθετη περίπτωση, σε έναν πραγματικό χρονοδρομολογητή χώρου πυρήνα, υπάρχει μία queue, ή οποία θα κρατάει τα σήματα και θα τα εκτελεί με τη σειρά που θα έχουν έρθει.

2. Κάθε φορά που ο χρονοδρομολογητής λαμβάνει σήμα SIGCHLD, σε ποια διεργασίαπαιδί περιμένετε να αναφέρεται αυτό; Τι συμβαίνει αν λόγω εξωτερικού παράγοντα (π.χ. αποστολή SIGKILL) τερματιστεί αναπάντεχα μια οποιαδήποτε διεργασία-παιδί;

Όταν ο χρονοδρομολογητής λαμβάνει σήμα SIGCHLD, βγάζει από τον χώρο πυρήνα την διεργασία που βρίσκεται αυτή τη στιγμή και προσθέτει την αμέσως επόμενη, που βρίσκεται στο queue. Αν μία διεργασία τερματιστεί αναπάντεχα, από μία εντολή SIGKILL, τότε ο χρονοδρομολογητής την κάνει remove από το queue και συνεχίζει πάλι στην αμέσως επόμενη.

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.

Το SIGALRM λαμβάνει χώρα όταν τελειώσει ένα κβάντο χρόνου, για να σταματήσει η διεργασία που έτρεχε και να πάρει θέση η διεργασία που έχει σειρά στην ουρά. Ωστόσο, χρειαζόμαστε και το SIGCHLD σε περίπτωση που τελείωσει κάποια διεργασία παιδί σε κάποιο κβάντο χρόνου. Τότε, θα σταλθεί το σήμα SIGCHLD στον χρονοδρομολογητή. Τότε αυτή τη διεργασία παιδί την κάνει remove και στο επόμενο κβάντο χρόνου ξεκίναει η επόμενη διεργασία παιδί που βρίσκεται στην ουρά. Ένω, άμα δεν έχει τελειώσει, την τοποθέτει στο τέλος της ουράς.

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;

Καθώς τρέχει το πρόγραμμα μας, όταν μέσω του cmd πληκτρολογήσουμε εμείς 'p', περιμένει η διεργασία που τρέχει τότε να τελειώσει στο κβάντο χρόνου και τότε "πηγαίνει" η χρονοδρομολόγηση στη διεργασία shell, η οποία και αναλαμβάνει να τρέξει τη συνάρτηση print.

```
Shell: issuing request...
Shell: receiving request return value...
Current PID: 306, name: shell
PID: 307, name: prog
```

Όπως φαίνεται, το shell παίρνει την εντολή και πάντα σαν current pid έχουμε το pid του shell.

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.

Θέλουμε να δεχόμαστε και να στέλνουμε σήματα όταν τελειώνει κάποιο κβάντο χρόνου, δηλαδή όταν δεν τρέχει κάποια διεργασία και ο χρονοδρομολογητής αλλάζει τη διεργασία που θα τρέξει. Οπότε, όταν το shell είναι στη μέση κάποιου αιτήματος, δε θέλουμε κάποιο εξωτερικό σήμα να διακόψει τη λειτουργία. Κάνουμε έτσι signal_disable στο TQ και μόλις εμφανιστεί ένα rq , τότε τα κάνουμε πάλι enable στο αμέσως επόμενο TQ.

3.3 Άσκηση 1.3

1. Περιγράψτε ένα σενάριο δημιουργίας λιμοκτονίας.

Προσθέτοντας αρκετές διεργασίες στον δρομολογητή (scheduler) και ορίζοντας αρκετές από αυτές σαν high (υψηλής προτεραιότητας), υπάρχει περίπτωση να μην τρέξει ή να αργήσει για αρκετά μεγάλο χρονικό διάστημα να τρέξει κάποια διεργασία, η οποία έχει οριστεί σαν low (χαμηλής προτεραιότητας).