ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολη ΗΜ&ΜΥ
Λειτουργικά Συστήματα
Άσκηση 4
Ακ. Έτος 2012-13

Παπαδιάς Σεραφείμ | ΑΜ: 03109193
Παπαδημητρίου Κων/νος | ΑΜ: 03108769
Ημερομηνία: 19/02/2013

Άσκηση 1.1 Σχεδίαση χρονοδρομολογητή κυκλικής επαναφοράς στο χώρο χρήστη

(α) Σύντομη περιγραφή της λειτουργίας του χρονοδρομολογητή.

Ο χρονοδρομολογητής χειρίζεται τις διεργασίες με χρήση των σημάτων SIGSTOP και SIGCONT. Αρχικά όλες οι διεργασίες βρίσκονται σε κατάσταση αναμονής, εκτός από την πρώτη. Όταν περάσει ένα κβάντο χρόνου ο scheduler στέλνει SIGSTOP στην τρέχουσα διεργασία για να τη σταματήσει και στη συνέχεια SIGCONT στην επόμενη για να συνεχίσει. Την λίστα με τις διεργασίες που βρίσκονται σε κατάσταση ready την υλοποιούμε με μία κυκλικά συνδεδεμένη λίστα. Έτσι οι διεργασίες ενεργοποιούνται κυκλικά, η μία μετά την άλλη. Ο χρονοδρομολογητής έχει δύο signal handlers. Ένα για το SIGALRM που έρχεται στο τέλος κάθε κβάντου χρόνου και ένα για το SIGCHLD που έρχεται κάθε φορά που μία διεργασία αλλάξει κατάσταση. Πιο συγκεκριμένα, όταν ο χρονοδρομολογητής πιάσει ένα SIGALRM, στέλνει ένα σήμα SIGSTOP στη τρέχουσα διεργασία. Αν όλα γίνουν καλά, αυτή θα αλλάξει κατάσταση και θα σταματήσει οπότε θα έρθει ένα σήμα SIGCHILD στη διεργασία του χρονοδρομολογητή. Τότε θα ενεργοποιηθεί ο handler για αυτό το σήμα. Ο handler του SIGCHILD εκτελεί τις εξής λειτουργίες:

- Αν η διεργασία που άλλαξε κατάσταση, ήταν σταματημένη και δέχτηκε SIGCONT, τότε απλά αγνοεί το SIGCHILD.
- Αν η διεργασία που άλλαξε κατάσταση, σταμάτησε από σήμα SIGSTOP, ενεργοποιεί την επόμενη στη λίστα αναμονής και προγραμματίζει ένα σήμα SIGALRM ώστε να έρθει σε ένα κβάντο χρόνου.
- Αν η διεργασία που άλλαξε κατάσταση, τερμάτισε τη λειτουργία της με exit(), την αφαιρεί από την λίστα και ενεργοποιεί με SIGCONT την επόμενη διεργασία στη λίστα. Προγραμματίζει όπως και πριν ένα σήμα SIGALRM να έρθει σε ένα κβάντο χρόνου.
- Αν η διεργασία που άλλαξε κατάσταση, πέθανε από κάποιο εξωτερικό σήμα (πχ καποιος της έστειλε SIGKILL), την αφαιρεί από τη λίστα.

(β) "Ψευδοκώδικας" του χρονοδρομολογητή.

Ο "ψευδοκώδικας" του κυρίου χρονοδρομολογητή:

- Αρχικοποίησε μια κυκλικά συνδεδεμένη λίστα διεργασιών.
- Αρχικοποίησε τη μεταβλητή nproc σε 0. (Εδώ κρατάμε τον αριθμό των διεργασιών στη λίστα αναμονής)
- Για κάθε στοιχείο στο argv[] δημιούργησε την αντίστοιχη διεργασία-παιδί, πρόσθεσέ τη στο τέλος της λίστας και αύξησε το nproc κατα 1. Κάθε διεργασία-παιδί πριν ξεκινήσει την εκτέλεσή της κάνει raise(SIGSTOP) για να σταματήσει.
- Εγκατέστησε δύο signal handlers. Ένα για το σήμα SIGALRM και ένα για το σήμα SIGCHLD. (Για τη λειτουργία του καθενός βλ. παρακάτω)
- Προγραμμάτισε να έρθει ένα SIGALRM σε tq δευτερόλεπτα (ένα κβάντο χρόνου).
- Ενεργοποίησε τη πρώτη διεργασία-παιδί στη λίστα αναμονής.
- *loop*: Περίμενε μέχρι να έρθει κάποιο σήμα.
- Πήγαινε στο *loop*.

Ο “ψευδοκώδικας” του SIGALRM handler:

- Στείλε στην τρέχουσα διεργασία SIGSTOP.

Ο “ψευδοκώδικας” του SIGCHLD handler:

- Αν η διεργασία που άλλαξε κατάσταση ήταν σταματημένη και συνέχισε λόγω σήματος SIGCONT τότε απλά αγνόησε το σήμα.
- Αν η διεργασία που άλλαξε κατάσταση σταμάτησε, τότε μάρκαρε τη τρέχουσα διεργασία της λίστας (είναι αυτή που έτρεχε και τώρα σταμάτησε) ως τελευταία και την επόμενή της ως τρέχουσα. Στη συνέχεια προγραμμάτισε ξανά να έρθει SIGALRM σε ένα κβάντο χρόνου και στείλε SIGCONT στη τρέχουσα διεργασία για να συνεχίσει την εκτέλεσή της.
- Αν η διεργασία που άλλαξε κατάσταση τερμάτισε “φυσιολογικά” τη λειτουργία της, τότε αφαίρεσε τη τρέχουσα διεργασία από τη λίστα αναμονής (είναι αυτή που τερμάτισε) και μείωσε την nproc κατά ένα. Αν nproc = 0 τότε τύπωσε ένα μήνυμα και τερμάτισε (δεν υπάρχει άλλη διεργασία στη λίστα αναμονής). Αλλιώς προγραμμάτισε ξανά να έρθει SIGALRM σε ένα κβάντο χρόνου, μάρκαρε την επόμενη διεργασία της τελευταίας ως τρέχουσα και στείλε της SIGCONT για να συνεχίσει τη λειτουργία της.
- Αν η διεργασία που άλλαξε κατάσταση τερμάτισε “βίαια” τη λειτουργία της (πχ εξαιτίας κάποιου σήματος SIGKILL), τότε εντόπισέ την στη λίστα (με το pid της) και αφαίρεσέ την από αυτή. Μείωσε την nproc κατα 1. Αν nproc = 0 τότε τύπωσε ένα μήνυμα και τερμάτισε (δεν υπάρχει άλλη διεργασία στη λίστα αναμονής). Αν η διεργασία που αφαίρεσες ήταν η τρέχουσα διεργασία, τότε μάρκαρε την επόμενη της τελευταίας ως τρέχουσα και στείλε στη τρέχουσα SIGCONT.

<u>Σημείωση:</u> Μαζί με τη παρούσα αναφορά επισυνάπτεται και κώδικας C. Εκεί εκτός της υλοποίησης των παραπάνω βρίσκονται ο ορισμός της λίστας και οι συναρτήσεις χειρισμού της.

<u>Ερωτησεις</u>

1.*<u>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;</u>*

Όταν έρθει ένα σήμα κατά την εκτέλεση της ρουτίνας ενός signal handler, αυτό αγνοείται προσωρινά, αφού γίνεται masked όταν ένας από τους δύο τρέχει. Στο χρονοδρομολογητή χώρου πυρήνα οι αλλαγές γίνονται μέσα στο χώρο του πυρήνα, οπότε τα σήματα που καταφτάνουν αναστέλλονται μέχρι να επιστραφεί ο χειρισμός στο χώρο χρήστη.

2.*<u>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;</u>*

Συνήθως το SIGCHLD προέρχονται από τη διεργασία που εκτελείται εκείνη τη στιγμή, η οποία δέχθηκε SIGSTOP επειδή τέλειωσε το κβάντο χρόνου της ή έκανε έξοδο. Επίσης μπορεί να προέρχεται από τη διεργασία, η οποία μόλις συνέχισε την εκτέλεσή της (επειδή είναι η επόμενη στη λίστα και δέχτηκε SIGCONT).
Γενικά SIGCHLD έρχεται όταν μια διεργασία-παιδί αλλάξει κατάσταση. Ο signal handler

του SIGCHLD ελέγχει το αντίστοιχο macro του status που πήραμε από τη waitpid (WIFCONTINUED, WIFSTOPPED, WIFEXITED ή WIFSIGNALED) και δρα ανάλογα. Αν δηλαδή μια διεργασία-παιδί πεθάνει εξαιτίας κάποιου εξωτερικού σήματος, την εντοπίζει στη λίστα και την αφαιρεί από αυτή.

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;*

Ένας scheduler που χρησιμοποιεί μόνο SIGALRM και δεν ελέγχει αν η διεργασία που προσπαθεί να σταματήσει όντως σταμάτησε, μπορεί να φτάσει σε κατάσταση όπου περισσότερες από μία διεργασίες προσπαθούν να εκτελεστούν ταυτόχρονα. Επίσης με τη χρήση SIGCHLD μπορούμε να ελέγξουμε διεργασίες που τερμάτισαν είτε από κανονική έξοδο είτε “βίαια” εξαιτίας κάποιου εξωτερικού παράγοντα.