Σχολή
Ηλεκτρολόγων Μηχανικών
και
Μηχανικών Υπολογιστών

Λειτουργικά Συστήματα

Γκούμας Βασίλης — 03113031
Ζαρίφης Νικόλαος — 03112178

1 Άσκηση 1η

1.1

Στο sigaction struct που έχουμε μέσα στην **install_signal_handlers** φτιάχνουμε μια μάσκα για τα σήματα SIGALARM ,SIGSTOP . Έτσι όταν εκτελείται ένας handler μπλοκάρονται τα σήματα εξαιτίας της μάσκας.

Ένας χρονοδομολογητής στο χώρο πυρήνα θα δούλευε με hardware interrupts, τα οποία οδηγούν στη μετάβαση απο χώρο χρήστη σε χώρο πυρήνα οπότε έχουμε εμφανέστερη ιεραρχία.

1.2

Κάθε φορά που λαμβάνει ένα σήμα SIGCHILD ο χρονοδρομολογητής μας δεν περιμένει το σήμα απο κάποια συγκεκριμένη διεργασία, έτσι όταν κάποια διεργασία στείλει σήμα είτε επειδή την σκοτώσαμε είτε επειδή έγινε pause , ο SIGCHILD handler θα πάρει το pid της κι θα την αφαιρέσει απο την λίστα (αν την σκοτώσαμε) μας αλλιώς θα εκκινήσει την επόμενη διεργασία.

1.3

Χρησιμοποιούμε 2 σήματα για λόγους συγχρονισμού γιατί τα σήματα δεν είναι ασφαλή κι έτσι θα μπορούσαμε παρόλο που στον κώδικα μας έχουμε την πρώτα το σήμα SIGSTOP , να πάει πρώτα το σήμα SIGCONT.Έτσι με δυο σήματα είμαστε σίγουροι ότι θα εκτελέστουν με την σωστή σειρά , αφού για να έρθει το SIGCHILD πρέπει πρώτα να έχει πάει στο παίδι το SIGSTOP.

2 Άσκηση 2η

2.1

Αυτό συμβαίνει γιατί το τύπωμα γίνεται όταν έχουμε δώσει την σκυτάλη στο shell , αφού τότε ξεκινάει η διαδικασία που παίρνει την εντολή μας , έτσι το shell θα βρίσκεται πάντα ως current. Ένας τρόπος για να μην συμβαίνει αυτό θα ήταν να λειτουργεί conquerently το shell κι έτσι θα τυπώνει αμέσως την διέργασια που εκτελείται. Βέβαια τότε θα είχαμε άλλα προβλήματα με την εργασία kill οπότε θα θέλαμε διαφορετική υλοποιήση.

2.2

Γιατί όταν λειτούργει ο φλοιός μπορεί κι εκτελεί κάποια διεργάσια μπορεί να έρθει κάποιο σήμα όπως SIGSTOP κτλπ κι το προγραμμά μας να σκάσει.Ο φλοιός επηρεάζει την λίστα που διατρέχουμε για τις διεργασίες έτσι θα μπορούσε πχ να σκοτώσουμε μια διεργάσια κι έτσι να κάνουμε free εναν κόμβο της λίστας που έκεινη την ώρα έλεγχε ο φλοιός.

3 Άσκηση 3η

Στην υλοποίηση μας χρησιμοποιήσαμε μία λίστα και η επιλογή της επόμενης διεργασίας έγινε ως εξης: Ψάχνουμε τη λίστα και εκτελούμε την πρώτη HIGH που βρίσκουμε. Άμα δε βρούμε καμία, έχουμε μόνο LOW, επομένως εκτελούμε οποιαδήποτε. Το μειονέκτημα σε σχέση με την υλοποίηση με δύο ουρές είναι ο χρόνος αναζήτησης της λίστας και πλεονέκτημα η εύκολη υλοποίηση, καθώς δεν χρησιμοποήσαμε εξωτερική βιβλιοθήκη για τις δομές των ουρών.

Αν είχαμε το shell ως χάμηλης προτεραίοτητας (ωστέ να μην μπορούμε να αλλάξουμε προταιρεώτητες) κι είχαμε ως HIGH διεργάσιες που θα αργούσαν πολύ να τερματίσουν , άλλες εργάσιες που είναι LOW θα έπρεπε να περιμένουν αρκετά
.

4 Προαιρετικές Ασκήσεις

4.1

Ο φλοιος δημιουργείται με την κλήση της sched_create_shell, η οποία εγκαθιδρύει την επικοινωνία του προγράμματος μας με την διεργασία shell, η οποία γίνεται μέσω pipes. Στη συνέχεια η sched_create_shell καλεί την do_shell για το fork της διεργασίας.

Στη συνέχεια το πρόγραμμα μας ”ακούει” για requests μέσω της shell_request_loop. Μόλις ληφθεί ένα request επεξεργάζεται μέσω της process_request(), όπου οι απαραίτητες δομές έχουν οριστεί στην βιβλιοθήη request.h.

Μέσα στην shell_request_loop φροντίζουμε να απενεργοποιήσουμε τους signal handlers καθώς ο χειρισμός των SIGALRM και SIGCHLD είναι ανεπιθύμητος, καθώς μπορεί να αλλάξει την κατατάσταση του κόσμου των διεργασιών όπως την γνώριζε το shell.

4.2

Μια υλοποίηση ενός μηχανισμού γήρανσης μπορεί να έχει ως εξής: κάθε ένα χρονικό διάστημα t αυξάνουμε τις προτεραιότητες όλων των διεργασιών κατα 1. πχ αν έχουμε μόνο τα επίπεδα HIGH-LOW κάθε t οι LOW γίνονται HIGH. Η λίστα των high εκτελείται με preemptive scheduling άρα είμαστε σίγουροι ότι θα εκτελεστεί στο άμεσο μέλλον. Η παραπάνω στρατηγική δουλεύει και αν έχουμε περισσότερα επίπεδα προτεραιοτήτων, πχ 0-255.

4.3

Το πρόβλημα με το σενάριο που περιγράφεται είναι ότι λόγω του αέναου βρόχου της M πριν απελευθερωθεί ο σημαφόρος, θα εκτελείται συνεχόμενα η M (starvation για τις H, L). Μία λύση είναι να εφαρμόσουμε ένα σχήμα γήρανσης όπως αυτό που περιγράφηκε παραπάνω, ώστε να αποφευχθεί το πρόβλημα που παρουσιάστηκε.