Βαΐα Νταφοπούλου 3050

Βίκτωρ Μεγήρ 3026

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ
REPORT

Περιεχόμενα:

1.Περιγραφη λειτουργιας χρονοδρομολογητη.

Η βασικη ιδεα του κωδικα που παρδιδεται ειναι η υλοποιηση ενος δικαιου χρονοδρομολογητη.Οι αλλαγες που καναμε στον αρχικο κωδικα στοχευουν στη δικαιη σειρα εκτελσης των διεργασιων.
Στη ουσια οταν μια νεα διεργασια δημιουργηθει ο διακομιστης χρονοδρομολογησης ειδοποιει τον πυρηνα με ενα μηνυμα και περιμενει τον πυρηνα να απαντησει στο μηνυμα.
Αφου ο διακομιστης επεξεργαστει τα δεδομενα της διεργασιας ειδοποιει τον πυρηνα ετσι ωστε ο τελευταιος να ειναι σε θεση να αποφασισει με δικαιο πλεον τροπο ποια διεργασια προκειται να εκτελεστει.Η αποφαση αυτη γινεται με κριτηριο την προτεραιοτητα .

Οι διεργασιες της ουρας ειναι χωρισμενες σε group στα οποια ο χρονοδρομολογητης ισομοιραζει τον χρονο εκτελεσης καθως και στις διεργασιες του καθε group.

2.Περιγραφη βασικων δομων και συναρτησεων.

Αρχικα καναμε καποιες αλλαγες στα πεδια της δομης διεργασιας δηλαδη προσθεσαμε τα procgrp ,proc_usage,grp_usage και fss_priority.Επειτα τα αρχικοποιησαμε στη συναρτηση do_start_scheduling.Το πεδιο procgrp παιρνει τιμη απο ενα μηνυμα που ο sched λαμβανει απο τον pm.Παρατηρησαμε οτι ο pm στελνει μηνυμα στον vfs ο οποιος του απανταει και ο pm μεσω της handle_vfs_reply διαχειριζεται το μηνυμα.Η handle_vfs_reply καλει την sched_start_user και αυτη καλει τη sched_inherit η οποια στελνει τελικα το μηνυμα.Οι αλλαγες στις συναρτησεις ειναι οι εξης:

sched_inherit: Στη συναρτηση αυτη εχουμε προσθεσει αλλο ενα ορισμα τυπου int στη δευτερη θεση.Στο σωμα της συναρτησης προσθεσαμε μια εντολη εκχωρησης χρησιμοποιώντας ενα νεο πεδιο SCHEDULING_MESSAGE το οποιο εχει οριστει στο include/minix/config.h.Αυτο το πεδιο παιρνει την τιμη του παραπανω int που αντιπροσωπευει το mp_procgrp της διεργασιας.Τελος αλλαξαμε το προτοτυπο της στο include/minix/sched.h

sched_start_user: Επειδη καλει την sched_inherit περασαμε ενα ακομη ορισμα ,το rmp->mp_procgrp.

do_start_scheduling: Προσθεσαμε μια εντολη που εκχωρει στο procgrp της διεργασιας το mp_procgrp του μηνυματος.Επισης ενα κομματι κωδικα που αρχικοποιει τα πεδια της καθε διεργασιας.Συγκεκριμενα για το grp_usage γινεται ο εξης ελεγχος: Ελεγχουμε το procgrp της τρεχουσας διεργασιας με καθε αλλης και αν ταυτιζονται εστω και με μιας τοτε το grp_usage της τρεχουσας παιρνει την τιμη της διεργασιας με το ιδιο procgrp.
Επισης δημιουργησαμε τις συναρτησεις check και count και τις καλουμε στην do_start_scheduling με στοχο να βρουμε το NG (number of groups) γιατι το χρησιμοποιουμε στον υπολογισμο της προτεραιοτητας .

do_noquantum: Σε αυτη τη συναρτηση ενημερωνονται τα πεδια της διεργασιας βασει των τυπων που δινονται οταν ληγει το κβαντο της.Επισης το οταν αλλαζει το grp_usage μιας διεργασιας αλλαζει το grp_usage των υπολοιπων διεργασιων στο ιδιο group.

schedule_process: Στελνει μηνυμα στον πυρηνα μεσω του οποιου στελνει τη διεργασια που ο πυρηνας θα βαλει στην ουρα.Εμεις προσθεσαμε σε αυτο το μηνυμα το fss_priority της διεργασιας βαζοντας ενα πεδιο int ως ορισμα. Η schedule_process καλει την sys_schedule η οποια καλει την kernelcall(SYS_SCHEDULE,&m). Στον πυρηνα η main καλει την system_init η οποια καλει τν map(SYS_SCHEDULE,do_schedule). Η do_schedule καλει την sched_proc που καλει την RTS_UNSET μεσω της οποιας μπαινει εν τελει στην ουρα η διεργασια. Καναμε τις εξης αλλαγες:

do_schedule: Δηλωσαμε μια νεα μεταβλητη τυπου int, την fss_priority. Βαλαμε μια εντολη εκχωρησης χρησιμοποιώντας ενα νεο πεδιο SCHEDULING_FSS_PRIO το οποιο εχουμε ορισει στο include/minix/com.h. Επειτα περασαμε το fss_priority ως ορισμα στο σημειο που καλειται η sched_proc.

sched_proc: Αυξησαμε τα ορισματα της με ενα int που εχει την τιμη του fss_priority. Επισης καναμε εναν ελεγχο για το fss_priority και εκει εκχωρουμε το fss_priority στο p->p_fss_priority ωστε να περασει σωστα στην RTS_UNSET ως ορισμα. Τ ο πεδιο p_fss_priority εχει οριστει στο kernel/proc.h.Αλλαξαμε το προτοτυπο της στο kernel/proto.h.

do_schedctl: Στο σημειο που καλει την sched_proc αλλαξαμε τα ορισματα βαζοντας το τελευταιο ισο με -1.

sys_schedule: Προσθεσαμε στην τελευταια θεση ενα ορισμα fss_priority τυπου int και στην συνεχεια στο πεδιο (SCHEDULING_FSS_PRIO) του μηνυματος m εκχωρω την τιμη του ορισματος αυτου. Εχουμε επισης αλλαξει το προτοτυπο της συναρτησης στο include/minix/syslib.h.

pick_proc: Ουσιαστικα εκει γινεται η επιλογη της διεργασιας που προκειται να εκτελεστει. Η αλλαγη που καναμε αφορα μονο την ουρα χρηστη, δηλαδη την εβδομη ουρα. Φτιαξαμε εναν αλγοριθμο με τον οποιο εντοπιζουμε την διεργασια με το μικροτερο fss_priority και αυτη ειναι που θα βγει απο την ουρα.

3. Αρχεια πηγαιου κωδικα

/usr/src/include/minix/config.h
/usr/src/include/minix/com.h
/usr/src/include/minix/libsys.h
/usr/src/include/minix/sched.h
/usr/src/lib/libsys/sched_start.c
/usr/src/lib/libsys/sys_schedule.c
/usr/src/servers/sched/schedule.c
/usr/src/servers/pm/schedule.c
/usr/src/servers/sched/schedproc.h
/usr/src/kernel/system/do_schedule.c
/usr/src/kernel/system/do_schedctl.c
/usr/src/kernel/system/proto.h
/usr/src/kernel/proc.c
/usr/src/kernel/proc.h
/usr/src/kernel/system.h
/usr/src/kernel/system.c

4. Εκσφαλματωση του κωδικα – Παρατηρησεις

Για τον ελεγχο της ορθης αποστολης και ληψης μηνυματων βαλαμε εντολες printf στα καιρια σημεια του κωδικα οπου στελνονται και λαμβανονται και ειδαμε οτι οντως η διαδικασια γινεται σωστα και στις δυο περιπτωσεις (Οι printf εχουν μπει σε σχολια ή εχουν αφαιρεθει).

Επιπροσθετα για να ελεγξουμε την ορθοτητα της χρονοδρομολγησης δημιουργησαμε ενα script οπως υποδεικνυοταν στην εκφωνηση και το τρεξαμε σε πολλα τερματικα (5).Στη συνεχεια παρατηρουσαμε σε καθε τερματικο τη συχνοτητα της εκτελεσης της εντολης echo του script.Το συμπερασμα μας ειναι οτι ο ρυθμος εκτελεσης της echo ειναι περιπου σταθερος σε ολα τα τερματικα.

Αυτο μας υποδηλωνει οτι η χρονοδρομολογηση γινεται σωστα διοτι ανεξαρτητα απο το ποσες διεργασιες φτασουν στον πυρηνα τελικα ο χρονος εκτελεσης θα ισομοιραστει μεταξυ τους.

==========