ΜΥΥ601 Λειτουργικά Συστήματα 2017

Εργαστηριακή Άσκηση 2 - *Υλοποίηση Δικαιης Χρονοδρομολόγησης στο MINIX 3.2.0*

1) Μεταφορά ταυτότητας οδηγού ομάδας διεργασίας

- Στο αρχείο **/usr/src/servers/pm/schedule.c**, κατά την επιστροφή της η συνάρτηση **sched_start_user( )** καλεί την **sched_inherit( )**, η οποία στέλνει μήνυμα στον *sched*.

- Στα ορίσματα της **sched_inherit( )** προσθέσαμε το `rmp->mp_procgrp`. Με αυτόν τον τρόπο το μήνυμα του *pm* στον *sched* περιλαμβάνει τον οδηγό ομάδας της διεργασίας.

- Χρειάστηκε να αλλάξουμε τη δήλωση της συνάρτησης **sched_inherit( )** (αρχείο **/usr/include/minix/sched.h**) και τον ορισμό της (αρχείο **/usr/src/lib/libsys/sched_start.c**) και να προσθέσουμε το επιπλέον πεδίο.

- Ο *sched* λαμβάνει αυτό το μήνυμα στη συνάρτηση **do_start_scheduling( )** του αρχείου **/servers/sched/schedule.c**. Εκεί το περνάμε στο πεδίο **procgrp** του **struct schedproc**.

- Στο αρχείο **/servers/sched/schedproc.h** προσθέσαμε το πεδίο **procgrp** για τον παραπάνω λόγο.

  2) Εισαγωγή, αρχικοποίηση και ενημέρωση βοηθητικών πεδίων για τον υπολογισμό προτεραιότητας διεργασιών χρήστη.

- Στο αρχείο **/servers/sched/schedproc.h** προσθέσαμε επιπλέον τα πεδία **proc_usage**, **grp_usage**, **fss_priority**.

- Στο ίδιο αρχείο, στην συνάρτηση **do_start_scheduling( )**, αρχικοποιήσαμε τα παραπάνω πεδία.

- Στην συνέχεια, στην συνάρτηση **do_noquantum( )** του ίδιου αρχείου, ενημερώνουμε τα βοηθητικά πεδία σύμφωνα με τον αλγόριθμο της δίκαιης χρονοδρομολόγησης.

  3) Μείωση του πλήθους των ουρών χρήστη και εφαρμογή της δίκαιης χρονοδρομολόγησης.

  - Αρχικά μειώσαμε το πλήθος των ουρών χρήστη τροποποιώντας το αρχείο **/usr/include/minix/config.h** και συγκεκριμένα:

```
#define NR_SCHED_QUEUES 9 / από 16 /
```

```
#define MAX_USER_Q 8 / από 0 /
```

- Στην συνάρτηση **schedule_process( )** του **/servers/sched/schedule.c** βρίσκουμε την διεργασία *χρήστη* (χρησιμοποιώντας έλεγχο για `priority == USER_Q`) με το ελάχιστο **fss_priority** και την δρομολογούμε καλώντας την **sys_schedule( )**. Με αυτό τον τρόπο εξασφαλίζουμε ότι μετά το επόμενο βήμα που θα περιγράψουμε, η διεργασία αυτή θα βρεθεί στην κεφαλή της ουράς διεργασιών στον πυρήνα.

- Αφού έχουμε κάνει το παραπάνω βήμα, καλούμε επαναληπτικά την **sys_schedule( )** για όλες τις υπόλοιπες διεργασίες που υπάρχουν στον πίνακα διεργασιών του *sched*. Οι διεργασίες αυτές μπαίνουν στην ουρά πίσω από την προηγούμενη (στο tail), και εφόσον μια διεργασία υπήρχε ήδη στην ουρά, γίνεται *dequeue* και απευθείας *enqueue* οπότε μπαίνει στο tail της ουράς.

- Προσέξαμε να συμπεριλάβουμε μόνο τις διεργασίες με `flags == IN_USE` για την αποφυγή σφαλμάτων.

  4) Δοκιμές σωστής λειτουργίας

  - Αρχικά τρέξαμε την εντολή `cd /usr/bin; sh sh;` (Το αρχείο sh είναι το script που μας δώθηκε στην εκφώνηση της άσκησης) σε 4 τερματικά και επαληθέυσαμε με τον ρυθμό εκτέλεσης της εντολής `echo` ότι ο χρόνος μοιράζεται ίσα στα groups.

- Επίσης τρέξαμε τα τεστ `cd /usr/src/test; make all; ./run` και όλα εκτός του 48 (αναμενόμενο) έδωσαν έξοδο **OK.**

Περιπτώσεις εκσφαλμάτωσης

1. Στο **/servers/sched/schedule.c** στην συνάρτηση **do_start_scheduling( )** προσθέσαμε ένα *printf( )* τυπώνοντας τον αριθμό του οδηγού ομάδας.
2. Στην **schedule_process( )** και τη **do_noquantum( )** του **/servers/sched/schedule.c** χρησιμοποιήσαμε αρκετά *printf( )* κατά το debugging.
3. Τρέχοντας τα τεστ `cd /usr/src/test; make all; ./run` βλέπαμε εάν όλα λειτουργούν όπως πρέπει.
4. Στο κάθε error που εμφανιζόταν, κοιτούσαμε στον πηγαίο κώδικα που υπάρχει το *printf( )* και ποια τα ορίσματα (error codes κλπ.) και έτσι αποκτούσαμε μια εικόνα για το που βρισκόταν το σφάλμα και τι το προκαλούσε.
5. Ειδική περίπτωση 1: Το σφάλμα `PM: SCHED denied taking over scheduling of (process name): (επιστεφόμενη τιμή της sched_start( ))`, το οποίο προκλήθηκε επειδή δεν είχαμε συμπεριλάβει το επιπλέον πεδίο στον ορισμό της συνάρτησης (βλ. (1) - δεύτερο bullet).

6. Ειδική περίπτωση 2: Το σφάλμα `PM: An error occurred when trying to schedule (process endpoint): (error code)` εμφανίστηκε αρκετά και το λύσαμε συμπεριλάμβάνοντας ελέγχους (κυρίως για το flag **IN_USE**) κατά την επαναληπτική κλήση της **sys_schedule( )**.

Μέλη Ομάδας

- **Χρήστος Ζώνιος**, *2194*
- **Νικόλαος-Μάριος Κόντος**, *2193*

ΥΓ: *Ευχαριστούμε το προσωπικό του μαθήματος και τον διδάσκοντα για την έγκαιρη καθοδήγηση όσον αφορά διευκρινίσεις και απορίες.*