Λειτουργικά εργασία 2 - Documentation

Μάρρας Ηλίας - ΑΜ: 1115201800111 - webmail: sdi1800111@di.uoa.gr

Αρχεία xv6 τα οποία έχω τροποποιήσει:

- **syscall.c**
- **syscall.h**
- **sysproc.c**
- **proc.c**
- **proc.h**
- **user.h**
- **defs.h**
- **usys.pl**

Αρχεία xv6 τα οποία έχω δημιουργήσει:

- **ps.c**
  Χρησιμοποιεί την κλήση συστήματος getpinfo(struct pstat *) για να εμφανίσει πληροφορίες σχετικά με τις ενεργές διεργασίες, σύμφωνα με τις προδιαγραφές της εργασίας. Αφού έχει γεμίσει η δομή struct pstat, εκτυπώνονται όλες πληροφορίες για κάθε ενεργή διεργασία μέσω αυτής της δομής.

- **pstat.h**
  Εδώ ορίζεται το struct pstat το οποίο κρατάει πίνακες μεγέθους NPROC που γεμίζουν μέσω της getpinfo για κάθε ενεργή διεργασία. Ακόμη, υπάρχει μια int μεταβλητή pnum που χρησιμεύει για την αποθήκευση του αριθμού των ενεργών διεργασιών που βρέθηκαν.

Κλήση συστήματος int setpriority (int num)

Για αρχή, στο syscall.c υπάρχει το πρότυπο της συνάρτησης για το system call και στο syscall.h αντιστοιχίζεται στο επόμενο νούμερο (23). Ύστερα, στο αρχείο sysproc.c υπάρχει το system call της setpriority το οποίο είναι υπεύθυνο για να λάβει το argument num μέσω της argint() και αφού ελέγχεται η εγκυρότητά του, καλεί την setpriority(η δήλωση της στο defs.h) που υλοποιείται στο αρχείο proc.c ως εξής: βρίσκει την διεργασία που τρέχει τη δεδομένη χρονική στιγμή μέσω της myproc() και αφού αποκτήσει πρόσβαση στην κρίσιμη περιοχή αλλάζει την προτεραιότητά της σε num. Επιστρέφει με 0 σε επιτυχημένη κλήση ή με -1 σε αποτυχημένη κλήση αντίστοιχα.

Κλήση συστήματος getpinfo(struct pstat *)

Όπως και με την setpriority υπάρχει το πρότυπο της συνάρτησης για το system call στο αρχείο syscall.c και αντιστοιχίζεται στο νούμερο (22) στο syscall.h. Στο αρχείο sysproc.c υπάρχει το system call της getpinfo το οποίο λαμβάνει μέσω της argaddr το argument pstat * και καλεί την getpinfo με αυτό το struct. Η getpinfo υλοποιείται στο αρχείο proc.c ως εξής: χρησιμοποιείται ένα struct local_ps προκειμένου να αντιγραφούν όλες οι πληροφορίες μιας ενεργής διεργασίας (έχω θεωρήσει ως ενεργή διεργασία τα state που είναι διαφορετικά απο UNUSED). Όσον αφορά την init διεργασία, λόγω του ότι δεν υπάρχει parent process για την συγκεκριμένη, ως ppid περνάω το 0 στο struct pstat. Ύστερα, προκειμένου να αντιγραφεί η πληροφορία από kernel level memory σε user level memory χρησιμοποιείται η συνάρτηση copyout() με το local_ps (που έχει ήδη γεμίσει) και το argument pstat (που διαβάζει εν τέλει η ps). Επιστρέφει με 0 σε επιτυχημένη κλήση ή με 1 σε αποτυχημένη κλήση αντίστοιχα.

Struct proc

Έχει προστεθεί η int μεταβλητή priority προκειμένου να αποθηκεύεται το priority ενός process. Ακόμη, στο αρχείο proc.c, στη συνάρτηση allocproc, με την αρχικοποίηση μιας νέας διεργασίας αρχικοποιείται και το priority της σε 10, σύμφωνα με τις προδιαγραφές της εργασίας.

Priority based Scheduler

Ο scheduler που έχω υλοποιήσει λειτουργεί ως εξής: Όπως και στον default scheduler υπάρχει το for(;;) loop έτσι ώστε να τρέχει συνεχώς και στη συνέχεια το πρώτο loop είναι υπεύθυνο για να βρει την διεργασία με την μέγιστη προτεραιότητα (δηλαδή προτεραιότητα μικρότερη της high_p). Ύστερα, έχουμε το δεύτερο loop το οποίο ψάχνει την διεργασία με την προτεραιότητα που έχουμε βρει (δηλαδή την "μεγαλύτερη") και δρομολογεί την διεργασία έτσι ώστε να εκτελεστεί. Σε κάθε loop κλειδώνει (και αντίστοιχα ξεκλειδώνει) το αντίστοιχο lock της διεργασίας που εξετάζεται.

Εκτέλεση

$ make
$ make qemu

$ priotest

```
$ priotest
Child pid 38, small, with priority 2 finished. Useless suCm:h 4i4l8d7 4p3i7d48
 19, small, with priority 2 finished. Useless sum: 448743748
Child pid 20, small, with priority 3 finished. Useless sum: 448743748
Child pid 39, small, with priority 3 finished. Useless sum: 448743748
Child pid 21, small, with priority 4 finished. Useless sum: 448743748
Child pid 40, small, with priority 4 finished. Useless sum: 448743748
Child pid 41, small, with priority 5 finished. Useless sum: 448743748
Child pid 22, small, with priority 5 finished. Useless sum: 448743748
Child pid 42, small, with priority 6 finished. Useless sum: 448743748
Child pid 23, small, with priority 6 finished. Useless sum: 448743748
Child pid 24, small, with priority 7 finished. Useless sum: 448743748
Child pid 43, small, with priority 7 finished. Useless sum: 448743748
Child pid 44, small, with priority 8 finished. Useless sum: 448743748
Child pid 25, small, with priority 8 finished. Useless sum: 448743748
Child pid 45, small, with priority 9 finished. Useless sum: 448743748
Child pid 26, small, with priority 9 finished. Useless sum: 448743748
Child pid 8, small, with priority 10 finished. Useless sum: 448743748
Child pid 27, small, with priority 10 finished. Useless sum: 448743748
Child pid 46, small, with priority 10 finished. Useless sum: 448743748
Child pid 28, small, with priority 11 finished. Useless sum: 448743748
Child pid 9, small, with priority 11 finished. Useless sum: 448743748
Child pid 47, small, with priority 11 finished. Useless sum: 448743748
Child pid 29, small, with priority 12 finished. Useless sum: 448743748
Child pid 10, small, with priority 12 finished. Useless sum: 448743748
Child pid 30, small, with priority 13 finished. Useless sum: 448743748
Child pid 11, small, with priority 13 finished. Useless sum: 448743748
Child pid 31, small, with priority 14 finished. Useless sum: 448743748
Child pid 12, small, with priority 14 finished. Useless sum: 448743748
Child pid 32, small, with priority 15 finished. Useless sum: 448743748
Child pid 13, small, with priority 15 finished. Useless sum: 448743748
Child pid 14, small, with priority 16 finished. Useless sum: 448743748
Child pid 33, small, with priority 16 finished. Useless sum: 448743748
Child pid 15, small, with priority 17 finished. Useless sum: 448743748
Child pid 34, small, with priority 17 finished. Useless sum: 448743748
Child pid 16, small, with priority 18 finished. Useless sum: 448743748
Child pid 35, small, with priority 18 finished. Useless sum: 448743748
Child pid 5, large, with priority 16 finished. Useless sum: 1818368003
Child pid 6, large, with priority 17 finished. Useless sum: 1818368003
Child pid 36, small, with priority 19 finished. Useless sum: 4C48h7i43l74d8
 pid 17, small, with priority 19 finished. Useless sum: 448743748
Child pid 18, small, with priority 20 finished. Useless sum: 448743748
Child pid 37, small, with priority 20 finished. Useless sum: 448743748
Child pid 7, large, with priority 18 finished. Useless sum: 1818368003
Child pid 4, large, with priority 20 finished. Useless sum: 1818368003
$
```

Εδώ να αναφέρω ότι όπως φαίνεται και στην πρώτη γραμμή στο παραπάνω screenshot, κάποιες φορές εκτυπώνονται σκουπίδια, μάλλον διότι το output κάποιου παιδιού εκτυπώνεται ταυτόχρονα με κάποιου άλλου.

$ ps

```
$ ps
PID     PPID    NAME    PRIORITY        SIZE    STATE
1       0       init    10              16384   SLEEPING
2       1       sh      10              20480   SLEEPING
3       2       ps      10              16384   RUNNING
$
```

$ usertests

```
OK
ALL TESTS PASSED
$
```