Λειτουργικά συστήματα, Εργασία 2, Περιττοί ΑΜ

ΠΑΝΑΓΙΩΤΗΣ ΠΕΤΡΑΚΟΠΟΥΛΟΣ - 1115 2021 00155

January 20, 2024

1 Εισαγωγή

Έχουν υλοποιηθεί όλες οι απαιτήσεις της εκφώνησης. Το make κάνει build χωρίς κανένα θέμα και το project τρέχει κανονικά. Τέλος, όλα τα tests κάνουν succeed. Γενικά ήταν αρκετά μικρές οι αλλαγές που χρειάστηκε να γίνουν, οπότε αυτή η αναφορά δύσκολα θα φτάσει τις δυο σελίδες.

2 Κλήσεις συστήματος

2.1 setpriority(int num)

Αρχικά προστέθηκε το member variable priority στο struct *proc* που βρίσκεται στο αρχείο *proc.h*

```
int priority;                // Process priority
```

και προστέθηκε η default τιμή 10 στην αρχικοποίηση του proc στις διαδικασίες fork και exec.

Στη συνέχεια δημιουργήθηκε το syscall setpriority(int num). Προστέθηκε το signature της διαδικασίας στα αρχεία *user.h* και $def s.h$. Επίσης προστέθηκε ένα παραπάνω entry στο αρχείο *usys.pl*. Δώθηκε ο αριθμός 22 στο syscall προσθέτοντας αυτό

```
#define SYS_setpriority 22
```

στο αρχείο *syscall.h* και τέλος προστέθηκε το prototype και το mapping στο αρχείο *syscall.c*:

```
...
extern uint64 sys_setpriority(void);

...
[SYS_setpriority] sys_setpriority,
```

Τέλος, ορίστηκε η συνάρτηση στο αρχείο *proc.c* και το αντίστοιχο syscall στο αρχείο *sysproc.c*. Η συνάρτηση απλά κάνει aquire το lock του process, αλλάζει το priority σε αυτό που ζητήθηκε απο τον χρήστη και κάνει release το lock του process. Το syscall διαβάζει τον αριθμό που δώθηκε ως όρισμα και καλεί τη συνάρτηση με αυτό.

2.2 getpinfo(struct pstat* stats)

Ορίστηκε το struct pstat ως εξής:

```
struct pstat
{
    int processesNum;

    struct pinfo activeProcesses[NPROC];
};
```

όπου το struct pinfo περιέχει διάφορες πληροφορίες για μια διεργασία.

Ακολουθήθηκε η ίδια διαδικασία με το προηγούμενο syscall για την δήλωση του, και επειδή δεν παρέχεται κάτι με την επανάληψη των παραπάνω θα παραληφθεί. Η συνάρτηση λαμβάνει ένα struct τύπου pstat και απλά διασχίζει τον πίνακα των υπάρχοντων διαδικασιών και για όσες έχουν state RUNNING ή RUNNABLE, δημιουργεί ένα entry τύπου pinfo στον πίνακα activeProcesses και αυξάνει το processesNum κατά ένα. Το syscall σε αυτή τη περίπτωση είναι λίγο πιο σύνθετο, αφού πρώτα διαβάζει τη διεύθυση που δώθηκε ως όρισμα και την κάνει cast σε struct pstat pointer. Στη συνέχεια δημιουργεί ένα επιπλέον struct pstat και με αυτό ως όρισμα καλεί τη συνάρτηση. Στο τέλος κάνει push το struct απο το kernel level στο user level και επιστρέφει το αποτέλεσμα.

3 Υλοποίηση προγράμματος χρήστη ps

Για την υλοποίηση του προγράμματος χρήστη ps, δημιουργήθηκε το αρχείο *ps.c*. Στη main αυτού το προγράμματος καλώ το syscall getpinfo και μετά δύο συναρτήσεις που όρισα για την παρουσίαση των στοιχείων που επιστρέφονται. Πρώτα καλώ την συνάρτηση printProcessesCount που απλά εκτυπώνει το πόσες ενεργές διεργασίες υπάρχου και μετά την συνάρτηση printActiveProcesses, η οποία εκτυπώνει τα στοιχεία του πίνακα activeProcesses. Τέλος χρειάστηκε να προσθέσω το αρχείο στο Makefile κάτω απο τα UPROGS ως εξής:

```
$U/_ps\
```

4 Υλοποίηση priority based scheduler

Για την Υλοποίηση του priority based scheduler, πρώτα όρισα την συνάρτηση

```
int findMinPriority()
```

η οποία εντοπίζει ποια είναι η χαμηλότερη τιμή priority που υπάρχει στις ενεργές διαδικασίες.

Ο scheduler σε κάθε iteration που κάνει, βρίσκει την χαμηλότερη τιμή priority που υπάρχει με τη χρήση της findMinPriority(), και στην συνέχεια περνάει πάνω απο όλη την λίστα των διεργασιών που υπάρχουν και δίνει ένα χρονομερίδιο με στυλ round robin σε όλες τις διαδικασίες που έχουν αυτή τη τιμή priority.