Λειτουργικά Συστήματα

3η Εργασία

Τσάμπρας Κωντσαντίνος

up1083865

* 1)

Γράφουμε το hello.c και απλά αλλάζουμε το μήνυμα που τυπώνεται στα logs και ακολουθούμε την ίδια διαδικασία.

Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές, στιγμιότυπο οθόνης, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

* 2)

Η δομή δεδομένων του linux task\_struct είναι ανάλογη με την δομή δεδομένων process του xv6, αφού και αυτή αντιπροσωπεύει μια διεργασία και αποθηκεύει όλες τις πληροφορίες που χρειάζεται μια διεργασία όπως η κατάσταση της διεργασίας, το αναγνωριστικό της διεργασίας, λεπτομέρειες διαχείρισης μνήμης, δείκτες αρχείων και πολλά άλλα. Όμως το xv6 είναι ένα εκπαιδευτικό kernel και έτσι η δομή process ήταν αρκετά απλή και δεν περιείχε πάρα πολλές πληροφορίες. Αντιθέτως η δομή task\_struct χρησιμοποιείται στα λειτουργικά linux που εξυπηρετούν τεράστιο πλήθος διαφορετικών λειτουργιών, οπότε είναι και πολύ πιο σύνθετες δομές.

Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές, στιγμιότυπο οθόνης, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

…

* 3)

Δημιουργούμε το module αρχικά συγγράφοντας το αρχείο list.c (παράλειψη το ότι δεν ονομάστηκε list-processes.c). Όπως και στο αρχείο που δίνεται, στην εναρκτήρια συνάρτηση τυπώνουμε το current process (το οποίο πρέπει να είναι το insmod που εισάγει το module στο kernel).   
Στην συνέχεια χρησιμοποιούμε το macro for\_each\_process χάρις στο οποίο η μεταβλητή p θα διατρέξει τον πίνακα των διεργασιών. Για κάθε μία από αυτές τις διεργασίες τυπώνουμε τον αριθμό τους (μέσα στην δικιά μας καταμέτρηση), το pid τους και το όνομά τους.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, ιστοσελίδα

Περιγραφή που δημιουργήθηκε αυτόματα

Στα αποτελέσματα (τα οποία βλέπουμε τυπώνοντας τα logs) βλέπουμε ότι πρώτη διεργασία με pid=1 είναι πάντα το system το οποίο είναι «πρόγονος» των υπολοίπων διεργασιών. Η λίστα είναι μεγάλη (176 διεργασίες) με τελευταία να εμφανίζεται η insmod, κάτι πού είναι λογικό, αφού αυτή η διεργασία δημιούργησε αυτήν την λίστα.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, ασπρόμαυρο

Περιγραφή που δημιουργήθηκε αυτόματα

…

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

* 4)
* Forking\_test.c

Αρχικά επεξεργάστηκα το forking.c (μετονομάστηκε σε forking\_test.c) προσθέτοντας επιπλέον χρόνο αρχικά για να γίνουν οι απαραίτητες αλλαγές στο list-processes.c (προσθήκη του νέου PID). Επίσης τροποποίησα το αρχείο ώστε οι forked processes (τα παιδιά) να μην κάνουν fork ώστε να μην γεμίζει το log με μηνύματα forking που δεν χρειάζονται, αφού μας ενδιαφέρουν μόνο τα παιδία της main process και όχι οι απόγονοι των παιδιών:

Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές, στιγμιότυπο οθόνης, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

Έτσι έχουμε αρχική αναμονή 60 δευτερολέπτων και έπειτα, κάθε 2 δευτερόλεπτα η main process δημιουργεί ένα νέο παιδί και τυπώνει το PID του παιδιού.

* list-children.c

Αρχικά έχουμε την σωστή τιμή για το PID

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, ιστοσελίδα, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

Στην συνέχεια έχουμε τον κώδικα που προσθέσαμε, στον οποίο απλά διατρέχουμε την διπλά διασυνδεδεμένη λίστα που είναι αποθηκευμένη στο children μέλος του task με το κατάλληλο macro. Για κάθε παιδί που βρίσκουμε τυπώνουμε το pid και το όνομα του process. Η συνάρτηση αυτή καλείται ανά τακτά χρονικά διαστήματα ανάλογα με τον timer που έχουμε ορίσει (εδώ πρακτικά κάθε δευτερόλεπτο).

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, ιστοσελίδα

Περιγραφή που δημιουργήθηκε αυτόματα

* Αποτέλεσμα:Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

  Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, έγγραφο

Περιγραφή που δημιουργήθηκε αυτόματα

…

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, ασπρόμαυρο, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Βλέπουμε δηλαδή ότι μετά την εισαγωγή του module στο Kernel, αρχίζει το module να τυπώνει την «αναφορά» ανά τακτά χρονικά διαστήματα.

Αρχικά η main process βρίσκεται ακόμα στην αναμονή οπότε το Module δεν «βρίσκει» κανένα παιδί.

Στην συνέχεια αρχίζουν να δημιουργούνται παιδιά κάθε 2 δευτερόλεπτα (περίπου κάθε δεύτερη αναφορά) τα οποία τυπώνονται στην λίστα.

Η εκτέλεση συνεχίζεται μέχρι να δημιουργηθούν και τα 15 παιδία και μετά εμφανίζουμε τα logs του kernel με την εντολή .