Λειτουργικά Συστήματα Εργασία 2

Πληροφορίες:

Κωνσταντίνος Μάριος Σγούρας: ΑΜ 1115202000178

Διαδικαστικά:

Όλος ο κώδικας της εργασίας βρίσκεται στο εσωτερικό του φακέλου. Τα αρχεία που έχουν αλλάξει είναι τα:

- defs.h
- exec.c
- sysproc.c
- trap.c
- vm.c

Για να δουλέψει ο κώδικας είτε τρέχουμε make qemu και βάζουμε τα προγράμματα που θέλουμε να τρέξουμε με το χέρι είτε καλούμε την make grade.

Σχεδιαστικές Επιλογές:

1. **Βήμα 1:** Η πρώτη άσκηση αναφέρεται στη δημιουργία vmprint() (αρχείο vm.c) για εκτύπωση του page table .Χρησιμοποιούμε μια βοηθητική συνάρτηση vmprint_rec() όπου με αναδρομική της χρήση εκτυπώνουμε τον πίνακα. Έγινε χρήση του κώδικα της freewalk() (με κάποιες αλλαγές). Άρα για κάθε level του pagetable αναδρομικά καλούμε την vmprint_rec(). Τέλος τοποθετήσαμε μια κλήση της vmprint() στο τέλος της συνάρτησης exec() στο αρχείο exec.c.
2. **Βήμα 2:** Στο δεύτερο βήμα σβήνουμε τον κώδικα της sys_sbrk() που κάνει allocate όλη τη φυσική μνήμη. Στη θέση του βάζουμε κώδικα που απλά μεγαλώνει το myproc()->sz κατα n και μετά απλά επιστρέφει το προηγούμενο μέγεθος.
3. **Βημα 3:** Για αυτό το βήμα δημιουργήσαμε τη συνάρτηση allocate_page() οπου δέχεται μια διεργασία και μια εικονική διεύθυνση και κάνει allocate την φυσική μνήμη που χρειάζεται. Αρχικά ελέγχουμε αν η εικονική μνήμη είναι στο σωστό εύρος (μικρότερη από την μνήμη που έχει αποδοθεί μέσω της sys_sbrk() και μεγαλύτερη από την αρχή της στοίβας του χρήστη). Στη συνέχεια χρησιμοποιούμε την PGROUNDDOWN για να στρογγυλοποιούμε την εικονική διεύθυνση στην χαμηλότερη διεύθυνση του ίδιου page. Έπειτα κάνουμε allocate την μνήμη μας. Αν υπάρχει χώρος συνεχίζουμε αλλιώς επιστρέφουμε 0. Στη συνέχεια γεμίζουμε το καινούργιο κομμάτι μνήμης με μηδενικά. Έπειτα επιχειρούμε να δημιουργήσουμε εγγραφή της συγκεκριμένης

μνήμης στον PageTable. Αν το επιτύχουμε επιστρέφουμε την θέση της μνήμης που δημιουργήσαμε αλλιώς επιστρέφουμε 0. Επίσης για να δουλέψει σωστά η συνάρτηση αλλάξαμε την mappages() και στην περίπτωση που έχει γίνει ήδη allocate δεν κάνουμε panic αλλα ξανακάνουμε map (δεν συμβαίνει αυτή η περίπτωση στα test, σωστότερη επιλογή θα ήταν να κάνουμε free το mem αλλα είναι εκτός των πλαισίων της άσκησης). Τέλος στην συνάρτηση usertrap δημιουργούμε μια περίπτωση οπου το r_scause() είναι ίσο με 13 η 15. Σε αυτή τη περίπτωση καλούμε την allocate_page και αν το αποτέλεσμά της είναι 0 δηλαδή επιστρέψει λάθος σκοτώνουμε τη διεργασία. Με αυτές τις αλλαγές το echo hi ήταν λειτουργικό. Στη συνέχεια φτιάξαμε την λειτουργικότητα των lazytests και usertests. Βασιστήκαμε στις οδηγίες της εκφώνησης. Αρχικά χειριστήκαμε τα αρνητικά ορίσματα στην sys_sbrk(). Στην περίπτωση αρνητικού ορίσματος λοιπόν κάναμε deallocate τις σελίδες που δεν χρειάζονταν χρησιμοποιώντας τη συνάρτηση uvmdealloc και ενημερώσαμε το myproc()->sz. Χρειάστηκε όμως να κάνουμε κάποιες αλλαγές στην uvmunmap που την καλεί η uvmdealloc. Παρατηρούμε οτι για κάθε σελίδα η uvmanmup κάνει ένα walk. Στη περίπτωσή μας το walk δεν εγγυάται οτι θα βρει PageTable Entry. Άρα στην περίπτωση που δεν βρει αγνοούμε τη συγκεκριμένη σελίδα. Στη συνέχεια αν το entry που βρήκε δεν είναι valid τότε δεν κάνουμε τα free που κάνει η συνάρτηση. Στη συνέχεια φροντίζουμε να τερματίζουμε την διεργασία αν η εικονική διεύθυνση που δώσαμε είναι μεγαλύτερη του sz της διεργασίας (αναφέραμε πως το καταφέραμε αυτό στην υλοποίηση της allocate_page). Έπειτα χειριζόμαστε την αντιγραφή μνήμης μέσω της fork(). Η fork χρησιμοποιεί την uvmcopy η οποία αντιγράφει το πατρικό PageTable. Κάθε φορά που δεν βρίσκει entry για μια σελίδα δεν την αντιγράφει. Αν το PageTable Entry που βρει είναι όχι valid τότε επίσης την αγνοεί. Αλλιώς κάνει allocate μνήμη και memmove (αντιγραφή) και mappages με σωστή διαχείριση αν υπάρχει λάθος. Στη συνέχεια φροντίζουμε την περίπτωση που η διεργασία περνάει μια έγκυρη εικονική διεύθυνση στην sys_read() ή sys_write() και δεν την έχουμε κάνει ακόμα allocate. Μέσω των συναρτήσεων filewrite και fileread οδηγούμαστε στις συναρτήσεις writei και readi οι οποίες με τη σειρά τους χρησιμοποιούν τις either_copyin και either_copyout που με τη σειρά τους χρησιμοποιούν τις copyin και copyout και μέσα τους καλούν την walkaddr(). Οι writei-readi συναρτήσεις γράφουν η διαβάζουν γράμματα από το inode. Οι συναρτήσεις either_copyin και either_copyout αντιγράφουν κάποιες εικονικές διευθύνσεις χρήστη ή πυρήνα και οι ncopyin και copyout αντιγράφουν δεδομένα από και προς τον πυρήνα. Η walkaddr όμως είναι εντέλι υπεύθυνη για την αναζήτηση μιας εικονικής διεύθυνσης. Για αυτό το λόγο αν η εικονική διεύθυνση δεν οδηγεί σε κάποιο entry η οδηγεί σε κάποιο invalid entry (η entry που ο χρήστης δεν έχει

δικαιώματα) φροντίζουμε να καλέσουμε την allocate_page και να κάνουμε allocate την απαραίτητη φυσική μνήμη καθώς και να επιστρέψουμε σωστά τον δείκτη προς αυτή (αν υπάρχει λάθος επιστρέφουμε 0). Στη συνέχεια φροντίζουμε να χειριζόμαστε της καταστάσεις εξάντλησης μνήμης σωστά. Όπως προαναφέραμε στην περιγραφή της allocate_page όταν υπάρχει πρόβλημα στην kalloc (δηλαδή υπάρχει έλλειψη μνήμης) τότε φροντίζουμε να επιστρέψουμε 0. Αξίζει να εξηγήσουμε οτι στην περιπτωση PageFault (δηλαδή στη συνάρτηση usertrap) όταν βρίσκουμε 0 στην επιστροφή της allocate_page τότε σκοτώνουμε τη διεργασία. Στην περίπτωση που η allocate_page μας επιστρέψει 0 στη συνάρτηση walkaddr επιστρέφουμε 0 το οποίο ύστερα από μια σειρά επιστροφών κάνει τις sys_read() και sys_write() να επιστρέψουν -1 (σωστή και απαιτούμενη λειτουργία για κάποια test). Τέλος πρέπει να φροντίσουμε να επιστρέψουμε λάθος αν η εικονική διεύθυνση βρίσκεται κάτω από τη στοίβα του χρήστη που όπως προαναφέραμε βρίσκεται δεν λειτουργικότητα στην allocate_page.