Λειτουργικά Συστήματα

Τμήμα Ζυγών Α/Μ

Άσκηση 3:

Υλοποίηση κοινής μνήμης στο λειτουργικό σύστημα xv6

Προθεσμία 23/12/17

(Παράδοση χωρίς ποινή έως 8/1/18 για όσους θέλουν να δουλέψουν στο διάστημα των διακοπών. Οι βοηθοί δεν έχουν καμία υποχρέωση να απαντούν στο Piazza μέσα στις γιορτές.)

1. Περιεχόμενο της Εργασίας

Στην εργασία αυτή καλείστε να εξοικειωθείτε με το σύστημα εικονικής μνήμης του λειτουργικού συστήματος xv6, και ύστερα να το επεκτείνετε προκειμένου να υποστηρίζεται η διαχείριση και χρήση κοινής μνήμης μεταξύ δύο ή περισσότερων διεργασιών. Η εργασία αποτελείται από τρία μέρη. Πρώτον, θα πρέπει να υποστηρίξετε την δημιουργία και διαγραφή διαμοιραζόμενων σελίδων για κοινή χρήση από δύο ή περισσότερες διεργασίες *(§1i shared pages)*. Δεύτερον, θα πρέπει να υλοποιήσετε, και να χρησιμοποιήσετε κατάλληλα, σημαφόρους, για τους σκοπούς του συγχρονισμού των διεργασιών κατά την πρόσβαση στα κοινά δεδομένα *(§1ii semaphores)*. Τέλος, θα πρέπει να δημιουργήσετε κατάλληλα προγράμματα εργασίας, τα οποία θα δημιουργούν και θα διαχειρίζονται το περιεχόμενο των shared pages *(§1iii workloads)*.

i. Shared Pages

Όπως έχει συζητηθεί στο μάθημα, το λειτουργικό σύστημα xv6 στερείται κάποιων λειτουργιών που υποστηρίζονται από τα περισσότερα σύγχρονα λειτουργικά συστήματα. Μία από αυτές είναι η υποστήριξη κοινής μνήμης μεταξύ των διεργασιών.

Κλήσεις συστήματος

Ζητείται να υλοποιήσετε δύο system calls που θα μπορεί να χρησιμοποιεί κάποιος χρήστης σε ένα δικό του πρόγραμμα εντός του xv6, ένα για τη δημιουργία μιας shared page και την κατάλληλη "τοποθέτησή" της στο address space της αιτούμενης διεργασίας, και ένα για τη διαγραφή μιας shared page από αυτό. Πιο συγκεκριμένα:

- Το system call **void* shmget(sh_key_t key)**, που είναι υπεύθυνο για τη δημιουργία και την τοποθέτηση του shared page στο virtual address space των επιμέρους διεργασιών, θα δέχεται ένα key, και θα επιστρέφει δείκτη στην αρχή του shared page.

- Αντίστοιχα, το system call **int shmrem(sh_key_t key)** που είναι υπεύθυνο για την αποδέσμευση της shared page σε επίπεδο εικονικής μνήμης, αλλά κι εν τέλει τη διαγραφή της, θα δέχεται ένα key το οποίο αναφέρεται σε μία shared page. Η κλήση θα επιστρέφει 0 αν η διεργασία ήταν η τελευταία που έκανε χρήση της shared page.

Αναφορά σε μια διαμοιραζόμενη σελίδα

Η αναφορά σε μία shared paged γίνεται μέσω ενός μοναδικού κλειδιού που της αντιστοιχεί. Τον ακριβή τύπο του κλειδιού θα τον ορίσετε εσείς, αλλά θα πρέπει να περιέχει/είναι τουλάχιστον δυαδικά δεδομένα 128bit (δηλ. ένα char[16] ή μεγαλύτερο), ώστε να επιτρέπει στις διεργασίες που θέλουν να επικοινωνήσουν με ασφάλεια να επινοήσουν ένα κρυπτογραφικό κλειδί που δεν σπάει με συμβατικά μέσα. Μπορείτε επίσης να ορίσετε τον τύπο κλειδιού έτσι ώστε το κάθε κλειδί να περιέχει και το μέγεθός του. Δηλαδή ο πυρήνας να ορίζει σαν σταθερά το μέγιστο μέγεθος του κλειδιού αλλά μια διεργασία να μπορεί να χρησιμοποιήσει και μικρότερο.

Προδιαγραφές - Περιορισμοί

Θεωρούμε ότι μπορούν να υπάρχουν το πολύ 32 shared pages, ενώ καθεμία shared page μπορούν να τη μοιράζονται το πολύ 16 διεργασίες, ανά πάσα στιγμή. Οι shared pages θα πρέπει να τοποθετούνται διαδοχικά από πάνω προς τα κάτω στις υψηλότερες διευθύνσεις του address space της διεργασίας (user-space), ακριβώς κάτω από τμήμα του address space που καταλαμβάνει ο πυρήνας (KERNBASE). Εάν δημιουργηθεί τρύπα λόγω διαγραφής κάποιας shared page από την διεργασία, τότε η επόμενη shared page που θα ζητηθεί θα πρέπει να τοποθετηθεί σε αυτήν την τρύπα (ή σε άλλη τρύπα, αν υπάρχει).

Δομές πυρήνα

Ο πυρήνας είναι υπεύθυνος για τον έλεγχο των περιορισμών (<=32 shared pages, <= 16 processes ανά shared page, η σελίδα δεν χρησιμοποιείται ήδη, κτλ.) και τη λήψη μέτρων, όποτε είναι απαραίτητο, αλλά, φυσικά, και για τη διαχείριση των shared pages σε επίπεδο φυσικής μνήμης. (**Υπόδειξη**: Για να δείτε πώς παίρνει ο πυρήνας φυσική μνήμη και πώς την απεικονίζει στην εικονική μνήμη μιας διεργασίας, διαβάστε τον κώδικα ξεκινώντας από την κλήση sbrk(). Η δική σας απεικόνιση μιας φυσικής σελίδας θα είναι παρόμοια, σε διαφορετικό σημείο του address space.)

Για αυτό το λόγο, θα πρέπει να δημιουργήσετε κατάλληλες δομές στον πυρήνα που περιέχουν τις απαραίτητες πληροφορίες για την ορθή διεκπεραίωση των παραπάνω, και θα πρέπει να φροντίσετε για το συγχρονισμό (με mutex locks) των λειτουργιών των δομών

αυτών όπου απαιτείται. Για παράδειγμα, ο πυρήνας θα πρέπει να κρατάει για κάθε shared σελίδα:

- τι κλειδί έχει
- ποιες διεργασίες την έχουν mapped και πού στο address space τους.
- … οτιδήποτε άλλο κρίνετε χρήσιμο.

Αντίστοιχα, ο πυρήνας θα πρέπει να έχει για κάθε διεργασία πληροφορία για το σε ποιες από τις πιθανές 32 θέσεις στην κορυφή του address space έχει mapped shared σελίδες και πού υπάρχουν τρύπες.

Οριστική διαγραφή μιας διαμοιραζόμενης σελίδας

Μια shared page θα διαγράφεται οριστικά μόνο όταν και η τελευταία διεργασία που έχει πρόσβαση σε αυτήν κάνει exit() ή αποφασίσει να τη διαγράψει με κλήση της *shmrem(...)*.

Υπόδειξη: Μελετήστε καλά τον κώδικα του xv6 και κάντε κατάλληλες αλλαγές έτσι ώστε όταν μια διεργασία κάνει exit() και δεν είναι η τελευταία που έχει πρόσβαση σε αυτήν, η shared page να μην διαγράφεται. Αντίστοιχα, σκεφτείτε ποια άλλα system calls αλλάζουν για να υποστηρίξουν σωστά τα shared pages. Το fork();

ii. Semaphores

Η χρήση κοινής μνήμης μεταξύ δύο ή περισσότερων διεργασιών εγκυμονεί κινδύνους, εάν δεν υπάρχει κάποιος μηχανισμός για τον συγχρονισμό των διεργασιών, με σκοπό την ορθή προσπέλαση των μεταξύ τους κοινών κομματιών μνήμης.

Δεδομένου του ότι το xv6 δεν υποστηρίζει κοινή μνήμη, δεν υπάρχει κάποιος μηχανισμός για προστασία της κοινής μνήμης (σε επίπεδο προγραμμάτων χρήστη), πέραν των locks (σε επίπεδο πυρήνα). Γι' αυτό το λόγο, θα υλοποιήσετε σημαφόρους ως μηχανισμό συγχρονισμού κατά την πρόσβαση στα shared pages. Πιο συγκεκριμένα θα πρέπει να οριστεί ένας νέος τύπος, ο τύπος του σημαφόρου - sem_t - για τον οποίο θα παρέχονται τρία system calls: για την αρχικοποίηση και για την υλοποίηση των λειτουργιών *UP* (ή αλλιώς *V*) και *DOWN* (ή αλλιώς *P*). Συγκεκριμένα, οι προδιαγραφές αυτών θα είναι οι εξής:

- **void sem_init(sem_t *sem, int value);**
- **void sem_up(sem_t *sem);**
- **void sem_down(sem_t *sem);**

(**Υπόδειξη**: Χρησιμοποιήστε τον κώδικα στο sleeplock.c σαν πρότυπο για την υλοποίηση των σημαφόρων σας.)

iii. Workloads

Προκειμένου να δοκιμάσετε τις υλοποιήσεις σας, θα πρέπει να δημιουργήσετε test programs, τα οποία θα δημιουργούν απλά δεδομένα/δομές σε shared pages, και θα συγχρονίζουν την πρόσβαση μέσω των semaphores. Κάνετε ό,τι νομίζετε για **διεξοδική** δοκιμασία της ορθότητας του κώδικά σας!

2. Διαδικαστικά

- Το πρόγραμμά σας θα πρέπει να γραφτεί σε C και να τρέχει στις μηχανές Linux workstations (linuxXY.di.uoa.gr ή linuxvmXY.di.uoa.gr) του τμήματος.
- Παρακολουθείτε την ιστοσελίδα του μαθήματος http://yanniss.github.io/k22/ για επιπρόσθετες ανακοινώσεις αλλά και την ηλεκτρονική-λίστα (η-λίστα) του μαθήματος στο https://piazza.com/uoa.gr/fall2017/k22/home .
- Αν ο σχεδιασμός των αρχείων σας είναι ερασιτεχνικός (π.χ. όλες οι δομές σε ένα αρχείο, λάθος διαχωρισμός header και .c files) θα υπάρξει βαθμολογική ποινή.

Τι πρέπει να παραδοθεί

1. Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματος σας και τις παρατηρήσεις σας.
2. Κατάλληλα τροποποιημένο Makefile του κώδικα του xv6.
3. ΄Ενα tar-file με όλη σας την δουλειά σε έναν κατάλογο που πιθανώς να φέρει το όνομα σας και θα περιέχει όλη σας την δουλειά δηλ. source files, header files, output files (αν υπάρχουν) και οτιδήποτε άλλο χρειάζεται.

΄Αλλες σημαντικές παρατηρήσεις

1. Οι εργασίες είναι **ατομικές**.
2. Αν και αναμένεται να συζητήσετε με φίλους/συμφοιτητές το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιαδήποτε μορφής) **δεν επιτρέπεται**. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικα **μηδενίζεται** στο μάθημα. Αυτό ισχύει για όσους εμπλέκονται ανεξάρτητα από το ποιος έδωσε/πήρε κλπ.
3. Αν πιθανώς κάποια (δευτερεύοντα!) κομμάτια του κωδικά σας προέλθουν από κάποια δημόσια πηγή, θα πρέπει να δώσετε αναφορά στη εν λόγω πηγή είτε αυτή είναι βιβλίο, σημειώσεις, Internet URL κλπ. και να εξηγήσετε πως ακριβώς χρησιμοποιήσατε την εν λόγω αναφορά.