Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Λειτουργικά Συστήματα (Κ22) / Περίοδος 2022-2023
2η Εργασία

Υποστήριξη Lazy Page Allocation στο xv6

Μία από τις τεχνικές που χρησιμοποιούνται σε λειτουργικά συστήματα σε συνδυασμό με hardware που υποστηρίζει σελιδοποίηση είναι η νωχελική (τεμπέλικη-lazy) κατανομή της μνήμης σωρού σε περιβάλλον χρήστη (user-space heap memory). Οι εφαρμογές στο xv6 αιτούνται από τον πυρήνα μνήμη σωρού χρησιμοποιώντας την κλήση συστήματος `sbrk()`. Στον υφιστάμενο πυρήνα, η `sbrk()` δεσμεύει φυσική μνήμη και την αντιστοιχίζει στο χώρο της εικονικής μνήμης της διεργασίας. Όμως, υπάρχουν προγράμματα που χρησιμοποιούν την `sbrk()` για να αιτηθούν μεγάλο μέγεθος μνήμης, χωρίς ποτέ να το χρησιμοποιήσουν ολόκληρο ή έστω το μεγαλύτερο τμήμα της (π.χ., υλοποίηση μεγάλων αραιών πινάκων). Για τη βελτιστοποίηση αυτής της περίπτωσης, οι εξελιγμένοι πυρήνες λειτουργικών συστημάτων εκχωρούν τη μνήμη περιβάλλοντος χρήστη νωχελικά (lazy). Η `sbrk()`, σε αυτή την περίπτωση, δεν εκχωρεί φυσική μνήμη, αλλά καταγράφει μόνο ποιές διευθύνσεις έχουν ανατεθεί. Όταν η διεργασία προσπαθήσει για πρώτη φορά να χρησιμοποιήσει οποιαδήποτε σελίδα μνήμης, η CPU παράγει ένα σφάλμα σελίδας (page fault), το οποίο ο πυρήνας διαχειρίζεται εκχωρώντας τις πραγματικές σελίδες φυσικής μνήμης, γεμίζοντας τις με μηδενικά και αντιστοιχίζοντας τις στις σελίδες εικονικής μνήμης

Βήμα 1 – Εκτυπώστε τον πίνακα σελίδων

Θα ξεκινήσετε με τη δημιουργία κώδικα που μπορεί να βοηθήσει αργότερα στο debugging. Υλοποιήστε μία συνάρτηση που εκτυπώνει τα περιεχόμενα ενός πίνακα σελίδων. Ορίστε τη συνάρτηση στο kernel/vm.c ως εξής:

```
void vmprint(pagetable_t pagetable);
```

Εισάγετε μία κλήση στην `vmprint()` στο τέλος της `exec()` στο kernel/exec.c για να τυπώσει τον πίνακα σελίδων για την πρώτη διεργασία χρήστη. Το αποτέλεσμα θα πρέπει να είναι όπως το παρακάτω:

```
page table 0x0000000087f6e000
 ..0: pte 0x0000000021fda801 pa 0x0000000087f6a000
 .. ..0: pte 0x0000000021fda401 pa 0x0000000087f69000
 .. .. ..0: pte 0x0000000021fdac1f pa 0x0000000087f6b000
 .. .. ..1: pte 0x0000000021fda00f pa 0x0000000087f68000
 .. .. ..2: pte 0x0000000021fd9c1f pa 0x0000000087f67000
 ..255: pte 0x0000000021fdb401 pa 0x0000000087f6d000
 .. ..511: pte 0x0000000021fdb001 pa 0x0000000087f6c000
 .. .. ..510: pte 0x0000000021fdd807 pa 0x0000000087f76000
 .. .. ..511: pte 0x000000002000200b pa 0x0000000080008000
```

Η πρώτη γραμμή εκτυπώνει τη διεύθυνση του ορίσματος της `vmprint()`. Κάθε γραμμή PTE (Page Table Entry) εκτυπώνει το δείκτη PTE στον κατάλογο σελίδων, το

pte και τη φυσική διεύθυνση (pa). Η εκτύπωση θα πρέπει να υποδεικνύει και το επίπεδο του καταλόγου σελίδων: στις εγγραφές στο ανώτατο επίπεδο προηγείται μόνο το string ‘ ..’, στο αμέσως κατώτερο επίπεδο ακόμη ένα string ‘ ..’ και στο κατώτερο επίπεδο άλλο ένα string ‘ ..’. Δεν πρέπει να εκτυπώνονται εγγραφές που δεν έχουν αντιστοιχηθεί (mapped). Στο συγκεκριμένο παράδειγμα ο ανώτατος ιεραρχικά κατάλογος σελίδων έχει αντιστοιχίσεις για τις εγγραφές 0 και 255. Το αμέσως κατώτερο επίπεδο από την εγγραφή 0 έχει μόνο το δείκτη 0 αντιστοιχισμένο και το κατώτατο επίπεδο για το δείκτη 0 έχει αντιστοιχίσει τις εγγραφές 0, 1, και 2.
Υποδείξεις:
Χρησιμοποιήστε τις μακροεντολές στο τέλος του αρχείου `kernel/riscv.h`

- Μπορείτε να δείτε τον τρόπο διάσχισης του ιεραρχικού πίνακα σελίδων από τη συνάρτηση `freewalk()`.
- Θα πρέπει να δηλώσετε τη συνάρτηση `vmprint()` στο αρχείο kernel/defs.h για να μπορείτε να τη χρησιμοποιήσετε στο kernel/exec.c.

Βήμα 2 – Απομακρύνετε την εκχώρηση μνήμης από την `sbrk()`

Θα πρέπει να διαγραφεί η εκχώρηση σελίδων από την κλήση συστήματος `sbrk()`, η οποία είναι η συνάρτηση `sys_sbrk()` στο αρχείο kernel/sysproc.c. Η κλήση συστήματος `sbrk(n)` μεγαλώνει το μέγεθος της μνήμης της διεργασίας κατά n bytes και στη συνέχεια επιστρέφει την αρχή της νεοανατεθείσας περιοχής (δηλαδή το παλιό μέγεθος). Η νέα `sbrk(n)` θα πρέπει απλώς να αυξάνει το μέγεθος της διεργασίας (`myproc()->sz`) κατά n και να επιστρέφει το παλιό μέγεθος. Δεν θα πρέπει να εκχωρεί μνήμη – προσοχή στην κλήση της `growproc()`.

Το αποτέλεσμα αυτής της τροποποίησης θα είναι παρεμφερές με το παρακάτω:

```
init: starting sh
$ echo hi
usertrap(): unexpected scause 0x000000000000000f pid=3
            sepc=0x0000000000001258 stval=0x0000000000004008
va=0x0000000000004000 pte=0x0000000000000000
panic: uvmunmap: not mapped
```

Το μήνυμα “`usertrap(): ...`” περιέχεται στο διαχειριστή user traps στο αρχείο kernel/trap.c. Αντιμετωπίζει ένα exception που δεν γνωρίζει πώς να το χειριστεί. Θα πρέπει να κατανοήσετε γιατί συμβαίνει αυτό το σφάλμα σελίδας. Η ένδειξη “`stval=0x0000000000004008`” υποδεικνύει ότι η εικονική διεύθυνση που προκάλεσε το σφάλμα σελίδας είναι η 0x4008.

Βήμα 3 – Νωχελική εκχώρηση (lazy allocation)

Τροποποιήστε τον κώδικα στο trap.c προκειμένου να αποκρίνεται σε σφάλμα σελίδας από το περιβάλλον χρήστη με την αντιστοίχιση μιας νεοανατεθείσας σελίδας φυσικής μνήμης στη διεύθυνση που προκάλεσε το σφάλμα και στη συνέχεια επιστρέφοντας στο περιβάλλον χρήστη για να αφήσει τη διεργασία να συνεχίσει να εκτελείται. Θα πρέπει να προσθέσετε τον κώδικά σας ως εναλλακτική περίπτωση ακριβώς πριν την κλήση στην `printf()` που εκτύπωσε το μήνυμα “`usertrap(): ...`”. Θα πρέπει να τροποποιήσετε και αλλού κώδικα του xv6 για να μπορέσει να εκτελεστεί η εντολή “`echo hi`”.

Υποδείξεις:

- Μπορείτε να ελέγξετε αν ένα σφάλμα είναι σφάλμα σελίδας αν η `r_scause()` έχει τιμή 13 ή 15 στην `usertrap()`.
- Η `r_stval()` επιστρέφει την τιμή του καταχωρητή stval της RISC-V CPU, ο οποίος περιέχει την εικονική διεύθυνση που προκάλεσε το σφάλμα σελίδας.
- Χρησιμοποιήστε κώδικα από την `uvmalloc()` στο kernel/vm.c, την οποία καλεί η `sbrk()` μέσω της `growproc()`.
- Χρησιμοποιήστε την `PGROUNDDOWN(va)` προκειμένου να στρογγυλοποιήσετε την εικονική διεύθυνση που προκάλεσε το σφάλμα σελίδας στην αρχή της σελίδας.
- Η κλήση της `uvmunmap()` θα προκαλέσει `panic()`. Τροποποιήστε την ώστε να διαχειρίζεται καλύτερα την περίπτωση όπου μερικές σελίδες δεν έχουν αντιστοιχιστεί.
- Χρησιμοποιήστε τη συνάρτηση `vmprint()` από το Βήμα 1 για να εκτυπώσετε το περιεχόμενο ενός πίνακα σελίδων.
- Μπορεί να χρειαστεί να κάνετε include τα αρχεία spinlock.h και proc.h αν εμφανιστεί το σφάλμα "`incomplete type proc`".

Στο τέλος θα πρέπει να εμφανιστεί η εντολή `echo hi` να δουλεύει, αντιμετωπίζοντας τουλάχιστον ένα σφάλμα σελίδας.

Επιπλέον για να αντιμετωπιστούν όλες οι περιπτώσεις θα πρέπει να αντιμετωπίσετε και τα εξής:

- Χειριστείτε αρνητικά ορίσματα στην κλήση της `sbrk()`.
- Τερματίσετε μία διεργασία αν προκαλέσει σφάλμα σελίδας σε διεύθυνση εικονικής μνήμης υψηλότερη από οποιαδήποτε ανατεθείσα μέσω της `sbrk()`.
- Χειριστείτε την αντιγραφή μνήμης πατέρα προς παιδί στην `fork()`
- Χειριστείτε την περίπτωση όπου μία διεργασία περνάει μία έγκυρη διεύθυνση από την `sbrk()` σε μία κλήση συστήματος όπως η `read()` ή `write()`, αλλά η μνήμη για τη διεύθυνση αυτή δεν έχει ακόμη ανατεθεί.
- Χειριστείτε καταστάσεις εξάντλησης μνήμης σωστά. Αν αποτύχει η `kalloc()` στον διαχειριστή σφαλμάτων σελίδας, τερματίσετε τη διεργασία
- Χειριστείτε σφάλματα στη μη έγκυρη σελίδα κάτω από τη στοίβα χρήστη.

Όταν ολοκληρώσετε την εργασία, ο πυρήνας θα πρέπει να περνάει όλα τα τεστ στα προγράμματα `lazytest` και `usertests`, δηλαδή:

```
$ lazytests
lazytests starting
running test lazy alloc
test lazy alloc: OK
running test lazy unmap...
tesst lazy unmap: OK
running test out of memory
test out of memory: OK
ALL TESTS PASSED
$ usertests
...
ALL TESTS PASSED
```

```
$
```

Μπορείτε επίσης να εκτελέσετε την εντολή `'make grade'`, η οποία θα ελέγξει και για την ορθή υλοποίηση της `vmprint()`.

Κώδικας

Χρησιμοποιήσετε τον κώδικα για την εργασία ως εξής:

```
$ git clone git://gallagher.di.uoa.gr/xv6-project-2022
Cloning into 'xv6-project-2022'...
...
$ cd xv6-project-2022
```

Το αποθετήριο (repository) xv6-project-2022 προσθέτει λίγη επιπλέον λειτουργικότητα σε σχέση με το κύριο. Για να δείτε τις αλλαγές που έχουν γίνει μπορείτε να δώσετε την εντολή

```
$ git log
```

Τα αρχεία που θα χρειαστείτε για αυτή την εργασία διανέμονται μέσω του συστήματος ελέγχου πηγαίου κώδικα git. Μπορείτε να βρείτε πληροφορίες για το git στο βιβλίο git ή σε άλλες δημόσιες πηγές. Το git επιτρέπει να διατηρείτε πληροφορία για όλες τις αλλαγές που έχετε κάνει στον κώδικα. Για παράδειγμα, αν τελειώσετε ένα μέρος της εργασίας και θέλετε να καταχωρήσετε τοπικά τις αλλαγές σας, μπορείτε να καταγράψετε (commit) τις αλλαγές σας μέσω της εντολής

```
$ git commit -am 'my solution for cow implementation project'
Created commit 60d2135: my solution for project 2 exercise 1
 1 files changed, 1 insertions(+), 0 deletions(-)
$
```

Ημερομηνία Παράδοσης: 22 Ιαν 2023
Τρόπος παράδοσης: υποβολή στο eclass, θα πρέπει να παραδοθεί ένα αρχείο tar με περιεχόμενο όλα τα σχετικά αρχεία.
Συνοδευτικό υλικό: τεκμηρίωση 3-4 σελίδων που να εξηγεί τον τρόπο με τον οποίο εργαστήκατε.
Υλοποίηση: η εργασία είναι ατομική.
Η εργασία θα εξεταστεί σε συμβατό xv6 emulator (Linux, Windows WSL) με πρόγραμμα που θα ανακοινωθεί μετά την ημερομηνία παράδοσης.