Καμάρης Άγγελος
sdi1900070

Για αυτή την εργασία δημιούργησα 1 πίνακα:

```
int reference_count[PHYSTOP/PGSIZE];
```

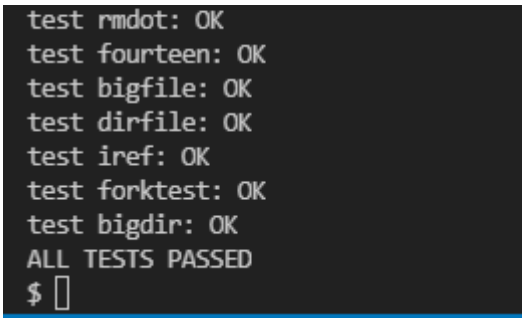
Και 2 συναρτήσεις, τις:

```
int cowfaulthandler(pagetable_t pagetable, uint64 va);
void addref(pagetable_t pagetable, uint64 pa);
```

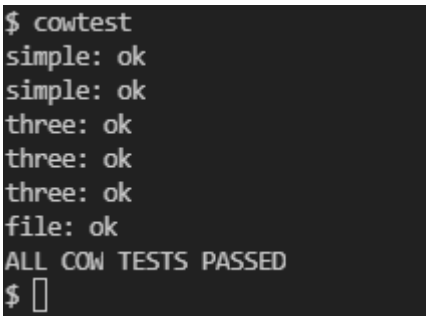
οι υπόλοιπες αλλαγές έγιναν σε ήδη υπάρχουσες συναρτήσεις στα αρχεία:
vm.c
kalloc.c
trap.c
vm.c

Το πρόγραμμα τρέχει με τις εντολές:
/xv6-project-2021$ make qemu
$cowtest
ή και
$usertests

Και μου βρίσκει σωστά αποτελέσματα και στα 2.
usertests:



cowtests:



Οι αλλαγές που έκανα στα αρχεία:

vm.c:

Συνάρτηση **uvmcopy:**

Άλλαξα στην γραμμή 315 τα flags ώστε να μην περιέχουν το κομμάτι της σελίδας που τους επιτρέπει να γράφουν σε αυτή και περνάω την σελίδα με αυτά τα flags στην συνάρτηση mappages ώστε να χρησιμοποιήσει την παλιά σελίδα pa χωρίς όμως να μπορεί να γράψει

σε αυτή το οποίο τελείωσα στην γραμμή 322. Τέλος έκανα την page table entry να μην μπορεί να γραφτεί και κάλεσα την συνάρτηση addref με δεδομένο το pa για να αυξήσει τον reference_counter αυτής της σελίδας μιας και πλέον πέρα από τον γονέα γίνεται referenced και από το παιδί, γραμμές 325-328.

Συνάρτηση **copyout**:

Στην γραμμή 360, καλώ τον cowfaulthandler για κάθε σελίδα, ώστε να αλλάζει σε κάθε σελίδα σελίδα που μπορεί να γίνει write, δίνοντάς του ως ορίσματα το pagetable που χρησιμοποιείται και το va0 το virtual address που δείχνει στην αρχή της εκάστοτε σελίδας.

kalloc.c:

Δημιούργησα έναν πίνακα μεγέθους `PHYSTOP/PGSIZE` ο οποίος θα χρησιμοποιηθεί για να περιέχει τον αριθμό που γίνεται reference κάθε σελίδα και για να είναι ατομικό για κάθε σελίδα το κλουβί χρησιμοποιώ το physicall address κάθε σελίδας που είναι μοναδικό ως προς το μέγεθος μια σελίδας, για να δημιουργήσω έναν μοναδικό αριθμό, γι αυτήν την σελίδα.

Συνάρτηση **kinit:**

Στην γραμμή 33 πρόσθεσα τη συνάρτηση memset, η οποία χρησιμοποιείται για να κάνει όλα τα κελιά του πίνακα 0.

Συνάρτηση **kfree:**

Στην γραμμή 60 αφαιρώ κατά 1 την τιμή του κελιού που έχει αριθμό `(uint64)pa/PGSIZE` στον refenrence_count, καθώς όταν καλούμε την kfree την καλούμε γιατί μια πρώην unwritable σελίδα μπορεί πλέον να γραφτεί, άρα δεν εξαρτάται από την pa σελίδα που είχε Ύστερα στην γραμμή 61 ελέγχω αν ο reference_count είναι μικρότερος ή ίσος με 0 και αν είναι διαγράφω την σελίδα pa που μου δόθηκε όπως γινόταν πριν την αλλαγή μου, κάνοντας παράλληλα το κελί στον πίνακα reference_counter πάλι 0 στην γραμμή 64 για να ξαναχρησιμοποιηθεί στο μέλλον, αφού αυτή η σελίδα δεν θα ξαναχρησιμοποιηθεί μέχρι να κληθεί kalloc γι'αυτό το pa.

Συνάρτηση **addref:**

Στις γραμμές 73-75 η συνάρτηση αυτή δέχεται σαν όρισμα το physical address μιας σελίδας και το χρησιμοποιεί για να βρει το κατάλληλο κελί και να του προσθέσει μια μονάδα. Καλείται όταν κάνουμε reference μια σελίδα στο **vm.c/uvmcopy**.

Συνάρτηση **kalloc:**

Στην γραμμή 91 προσθέτω μια μονάδα στον reference_count που χρησιμοποιεί σαν όρισμα την `(uint64)` τιμή του r προς το μέγεθος μια σελίδας που είναι ίδια με αυτή του physical address μιας σελίδας. Κατά τα άλλα η συνάρτηση κάνει ότι έκανε και πριν, δεσμεύει χώρο για μια σελίδα και επιστρέφει τον δείκτη σε αυτό τον χώρο.

defs.h:

Στην γραμμή 66 δηλώνω την συνάρτηση addref που χρησιμοποιείται στην **vm.c/uvmcopy**.

Στην γραμμή 149 δηλώνω την συνάρτηση **cowfaulthandler** αφού χρησιμοποιείται από την **trap.c/usertrap** και από την **vm.c/copyout**.

Στην γραμμή 175 δηλώνω την συνάρτηση walk η οποία καλείται και από την **trap.c/cowfaulthandle** πέρα από τις συναρτήσεις στην **vm.c**.

trap.c:

Συνάρτηση **usertrap:**

στην γραμμή 98 πρόσθεσα ένα κριτήριο αν ο λόγος που κλήθηκε η usertrap έχει την τιμή 15. Αν ισχύει αυτό σημαίνει ότι η σελίδα δεν μπορεί να γραφτεί άρα παίρνω το virtual address της σελίδας της διεργασίας στην γραμμή 100, ελέγχω αν αυτό είναι μέσα στο μέγεθος της σελίδας στις γραμμές 101-102, και καλώ την **trap.c/cowfaulthandler** με ορίσματα την σελίδα της διεργασία και το virtual address που δείχνει στην αρχή της σελίδας στις σελίδες 104-105.

Συναρτηση **cowfaulthandler:**

Γραμμές 32-59. Η συνάρτηση δέχεται ως ορίσματα μια σελίδα και την virtual address στην αρχή αυτής της σελίδας. ελέγχει αν η va είναι σωστή και αν είναι παίρνει το page table entry της σελίδας καλώντας την vm.c/walk. Αν αυτό δεν είναι άδειο, ελέγχει αν είναι valid, αν επιτρέπει χρήση από το user και αν ισχύουν και τα 2 καλεί την **kalloc.c/kalloc**, για να δεσμεύσει μνήμη για την σελίδα που θα δημιουργηθεί.από το page table entry δημιουργεί το physical address το οποίο δείχνει στην σελίδα που δεν μπορεί να γραφτεί και μια μεταβλητή flags ή οποία είναι όλα τα flags του page table entry μαζί με το flag που του επιτρέπει να γραφτεί. Ύστερα αντιγράφει τα περιεχόμενα του physical address στην σελίδα που δημιούργησε και κάνει το page table entry να δείχνει σε αυτή την σελίδα με τα καινούργια flags. Τέλος καλώ την **kalloc.c/kfree** στην παλιά σελίδα μιας και πλέον δεν γίνεται referenced. Σε περίπτωση σφάλματος επιστρέφεται -1 αλλιώς επιστρέφεται 0.

!Προσπάθησα να μείνω πιστός στον αρχικό κώδικα όσο γίνεται, προσθέτοντας όσο λιγότερες γραμμές γίνεται για να τρέξει. Τον δοκίμασα σε περιβάλλον WSL από υπολογιστή windows και ενώ η cowtest τελειώνει σε περίπου 10 δευτερόλεπτα, η usertests χρειάζεται κάτι παραπάνω από 3 λεπτά. Σε πιθανότητα που δεν λειτουργήσει η usertests δοκιμάστε να κάνετε make clean και να την ξανατρέξετε. Σας Ευχαριστώ για τον χρόνο σας.