**Λειτουργικά Συστήματα Υπολογιστών**

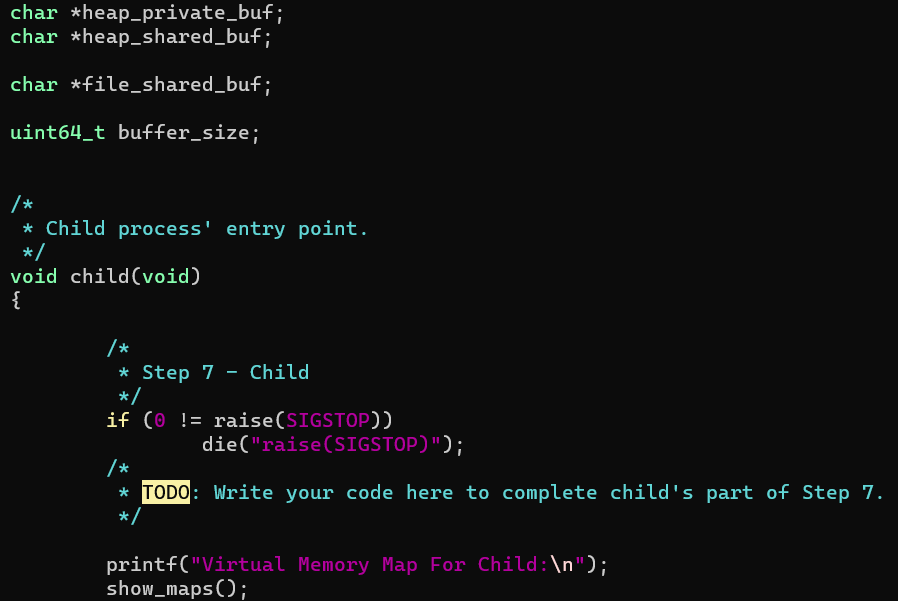
*Αναφορά στην 4η Εργαστηριακή Άσκηση*

Αλέξανδρος Σκούρας, 03120105

Ιωάννης Τσαντήλας, 03120883

*Εξάμηνο: Εαρινό 2022-23*

**1.1 Κλήσεις συστήματος και βασικοί μηχανισμοί του ΛΣ για τη διαχείριση της εικονικής μνήμης (Virtual Memory – VM)**



Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

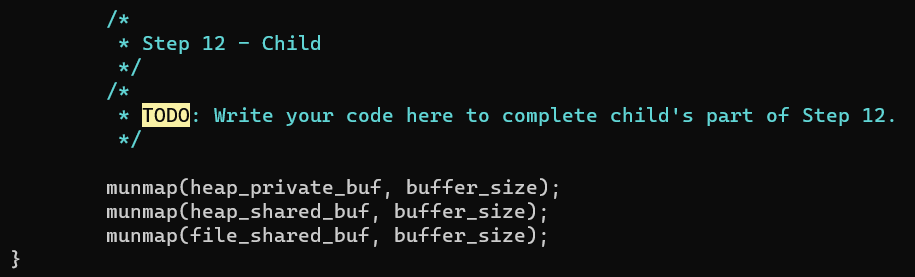
Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα



Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

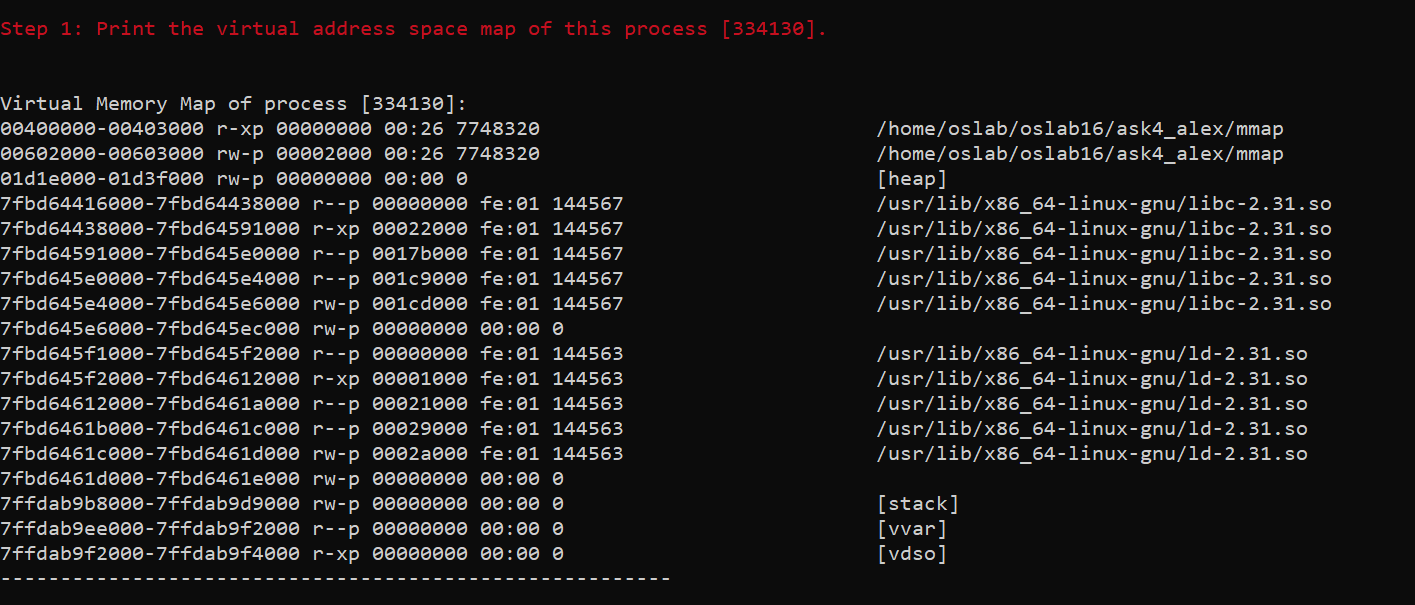
Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

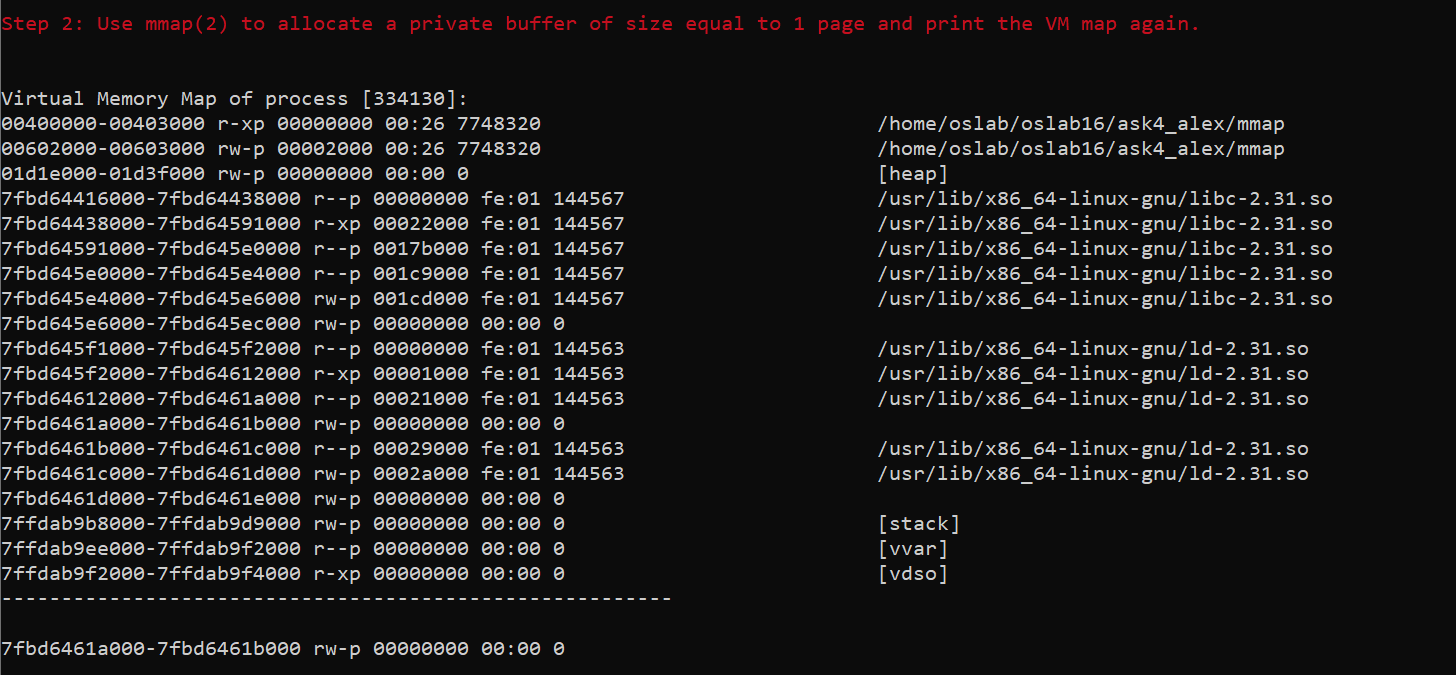
Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

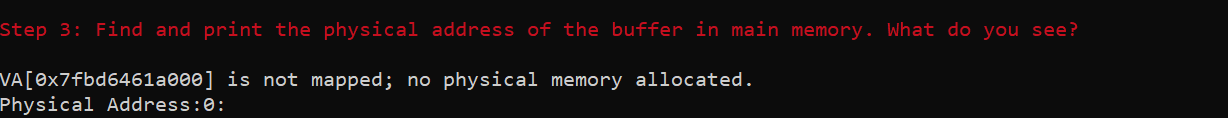
**Βήμα 1:**Τυπώνουμε τον χάρτη εικονικής μνήμης της τρέχουσας διεργασίας με χρήση της βοηθητικής συνάρτησης show\_maps().



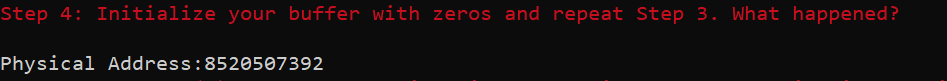
**Βήμα 2:**Δεσμεύουμε τον heap private buffer μεγέθους μιας σελίδας χρησιμοποιώντας την κλήση συστήματος mmap(), ξανατυπώνουμε τον χάρτη εικονικής μνήμης με τη show\_maps() και εντοπίζουμε τον χώρο που δεσμεύσαμε μέσω της show\_va\_info().



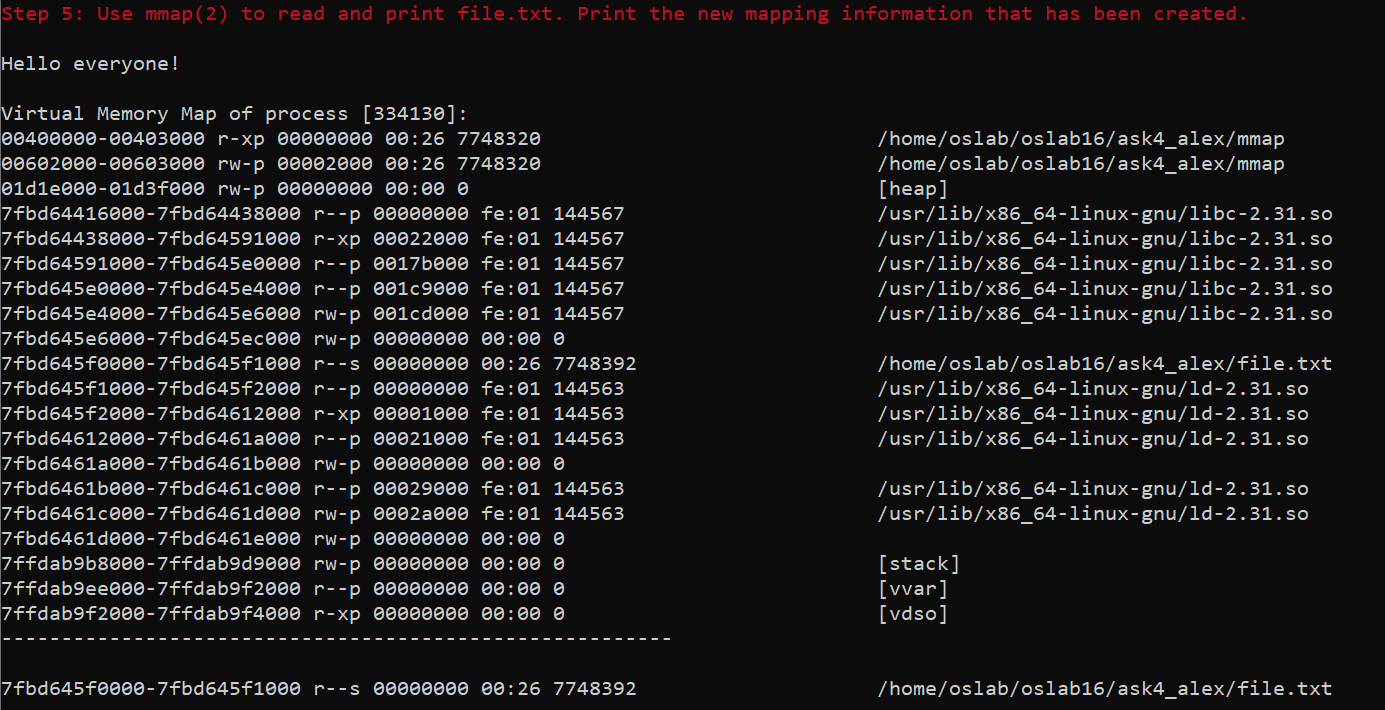
**Βήμα 3:**Επιχειρούμε να τυπώσουμε τη φυσική διεύθυνση μνήμης στην οποία απεικονίζεται η εικονική διεύθυνση του buffer μέσω της βοηθητικής συνάρτησης get\_physical\_address. Παρατηρούμε πως δεν έχει γίνει η απεικόνιση της εικονικής μνήμης σε φυσική. Κάτι τέτοιο είναι λογικό καθώς η φυσική μνήμη δεσμεύεται όταν πρόκειται να την προσπελάσουμε διαφορετικά μπορεί να καλύπταμε μέρος της μνήμης το οποίο εν τέλει να μην χρησιμοποιούσαμε ποτέ.



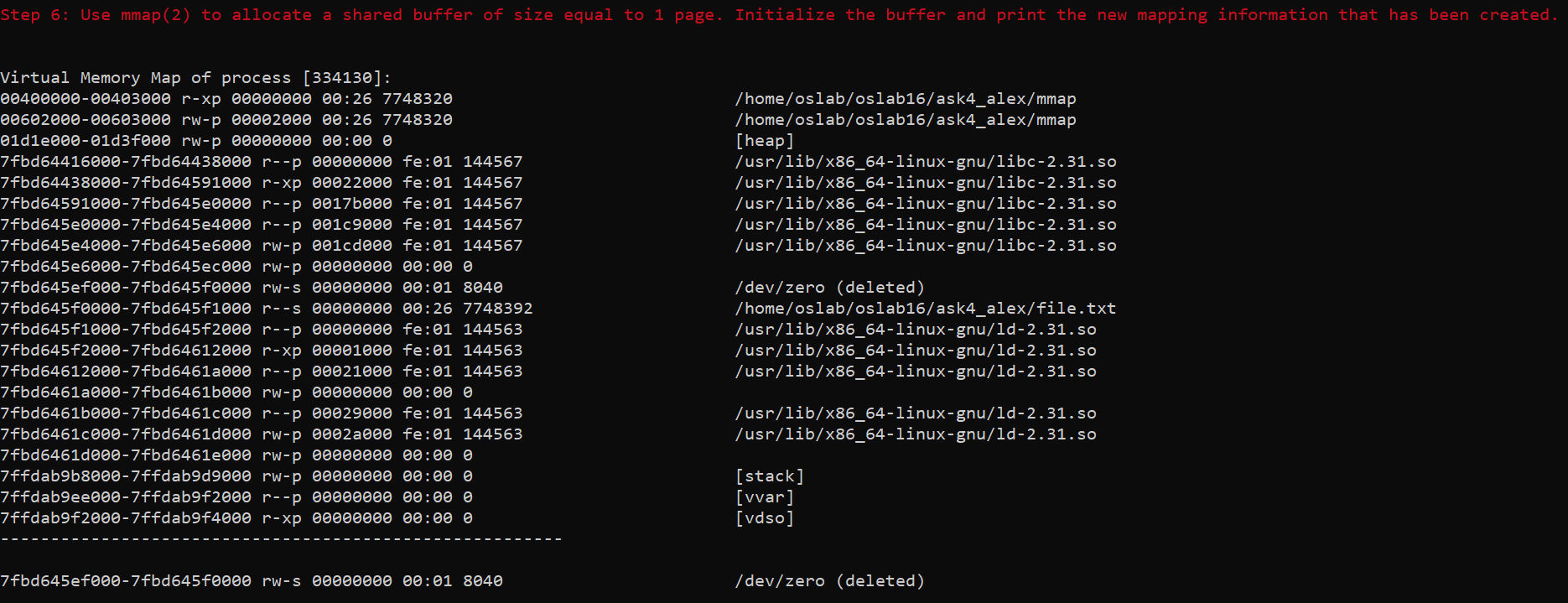
**Βήμα 4:**Γεμίζουμε τον buffer με μηδενικά δηλαδή προσπελαύνουμε την μνήμη με αποτέλεσμα πλέον να γίνεται η απεικόνιση της εικονικής σε φυσική.



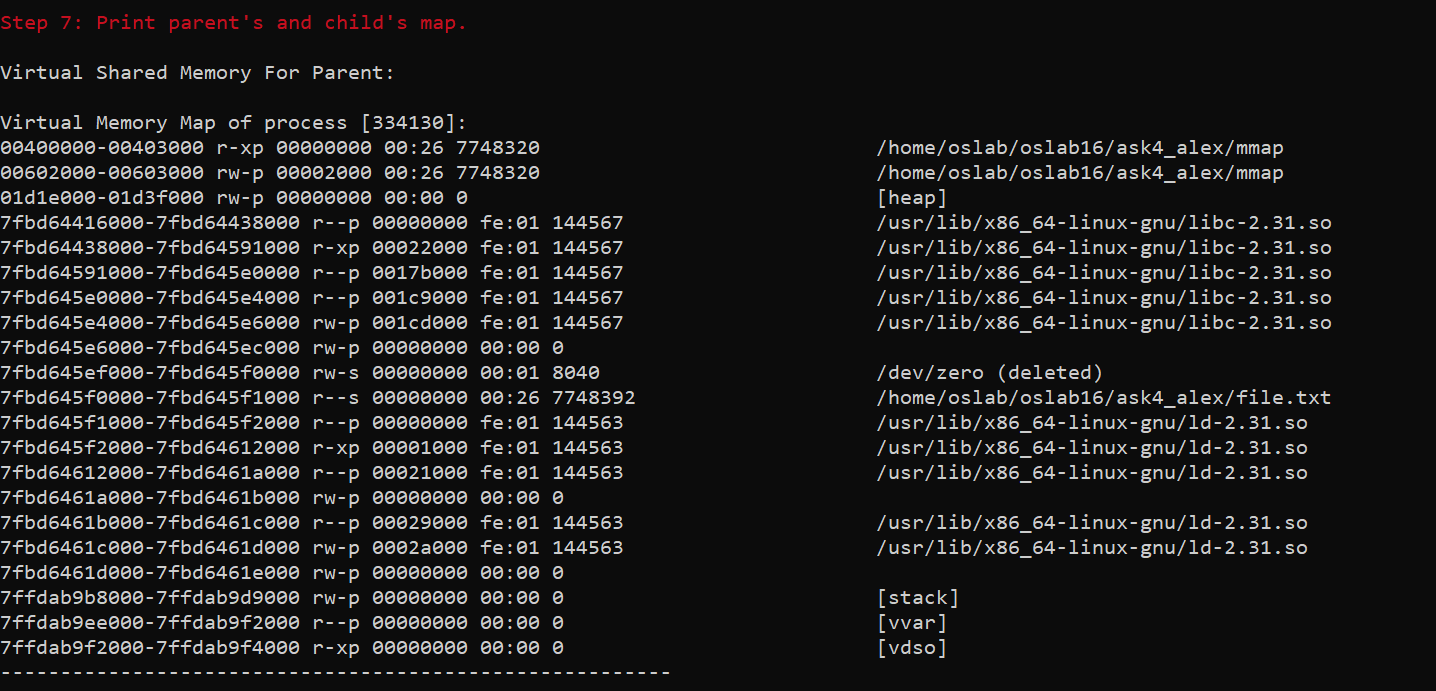
**Βήμα 5:**Με χρήση της mmap() δεσμεύουμε τον file shared buffer μεγέθους μίας σελίδας και απεικονίζουμε το αρχείο file.txt στον χώρο διευθύνσεων της διεργασίας μας τυπώνοντας παράλληλα το περιεχόμενό του.

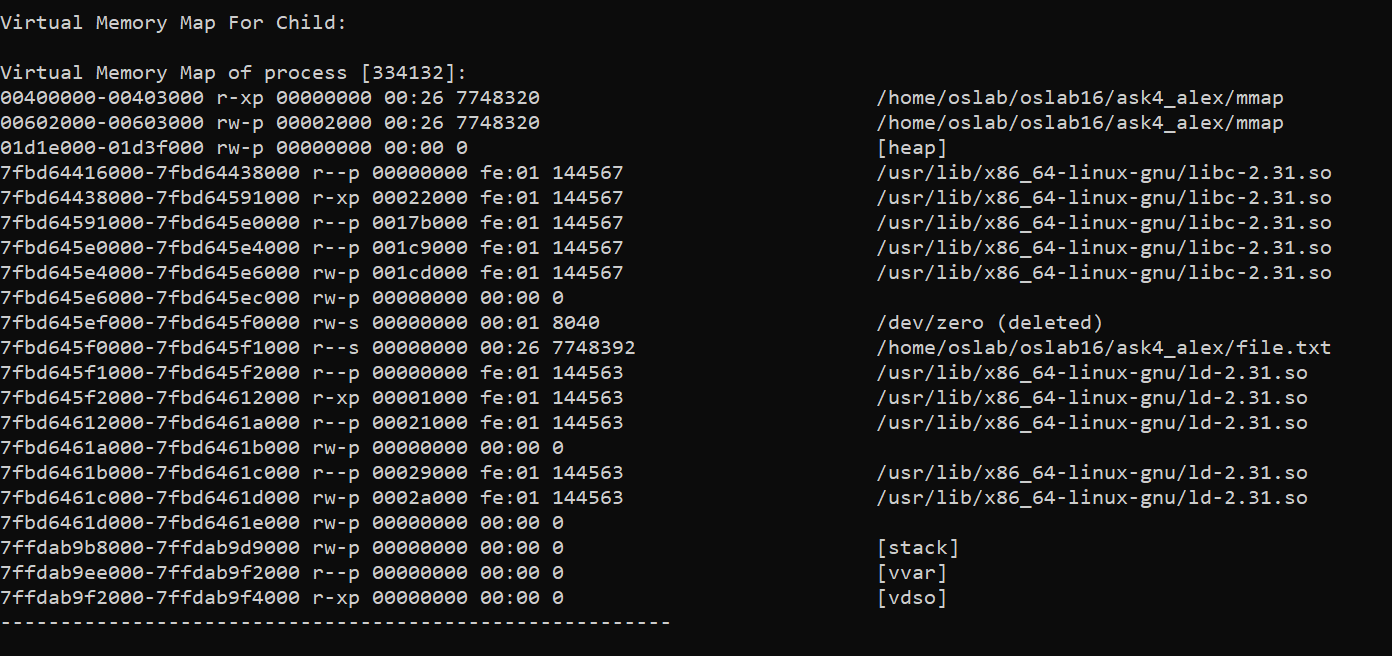


**Βήμα 6:**Δεσμεύουμε τον heap shared buffer(μοιραζόμενος μεταξύ των διεργασιών) μεγέθους μιας σελίδας μέσω της mmap(), τυπώνουμε τον χάρτη μνήμης και εντοπίζουμε την νέα απεικόνιση σε αυτόν.

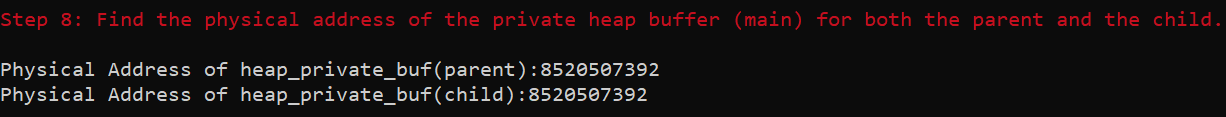


**Βήμα 7:**Τυπώνουμε τον χάρτη εικονικής μνήμης της διεργασίας παιδί και της διεργασίας πατέρα και παρατηρούμε πως είναι ίδιοι καθώς όπως περιμέναμε μέσω της fork δημιουργείται ένα αντίγραφο της αρχικής διεργασίας που κληρονομεί τα στοιχεία της.

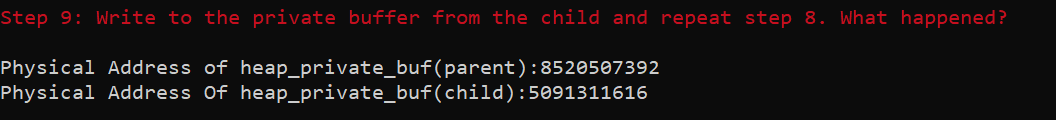




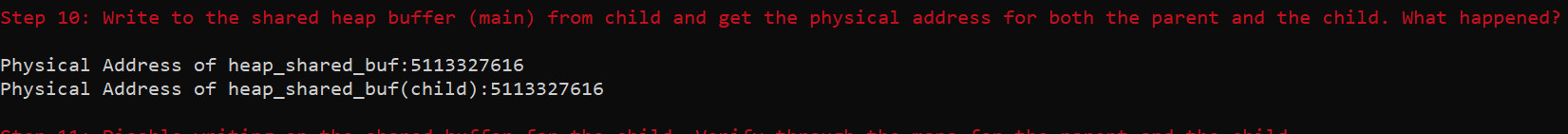
**Βήμα 8:**Τυπώνουμε την φυσική διεύθυνση μνήμης του private buffer για την διεργασία παιδί και για την διεργασία πατέρα και παρατηρούμε ότι έχουν την ίδια τιμή.



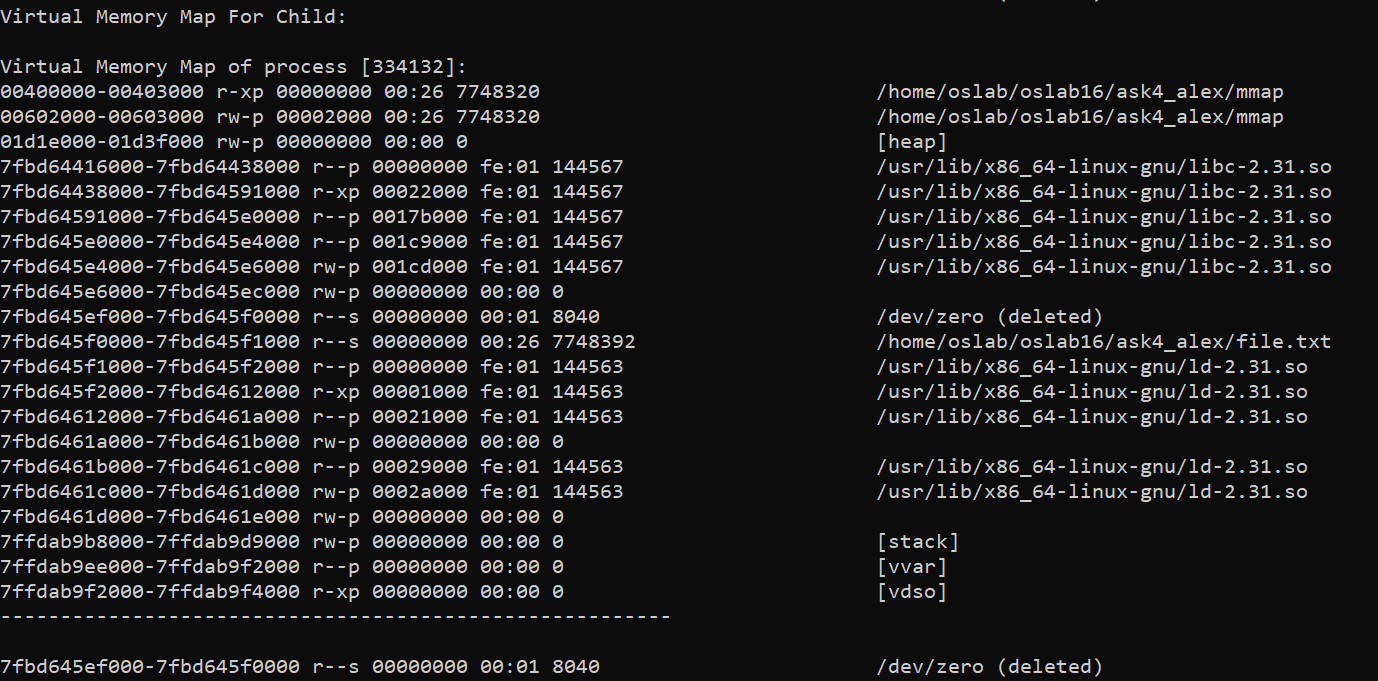
**Βήμα 9**:Γράφουμε στον private buffer από την διεργασία παιδί και παρατηρούμε ότι η απεικόνιση για την φυσική μνήμη του παιδιού αλλάζει. Αυτό συμβαίνει λόγω της χρήσης του flag MAP\_PRIVATE που δεν επιτρέπει στην διεργασία πατέρα και στην διεργασία παιδί να γράψουν στο ίδιο σημείο της φυσικής μνήμης καθώς έτσι η μια διεργασία θα μπορούσε να αλλάξει τα πράγματα που γράφει η άλλη.

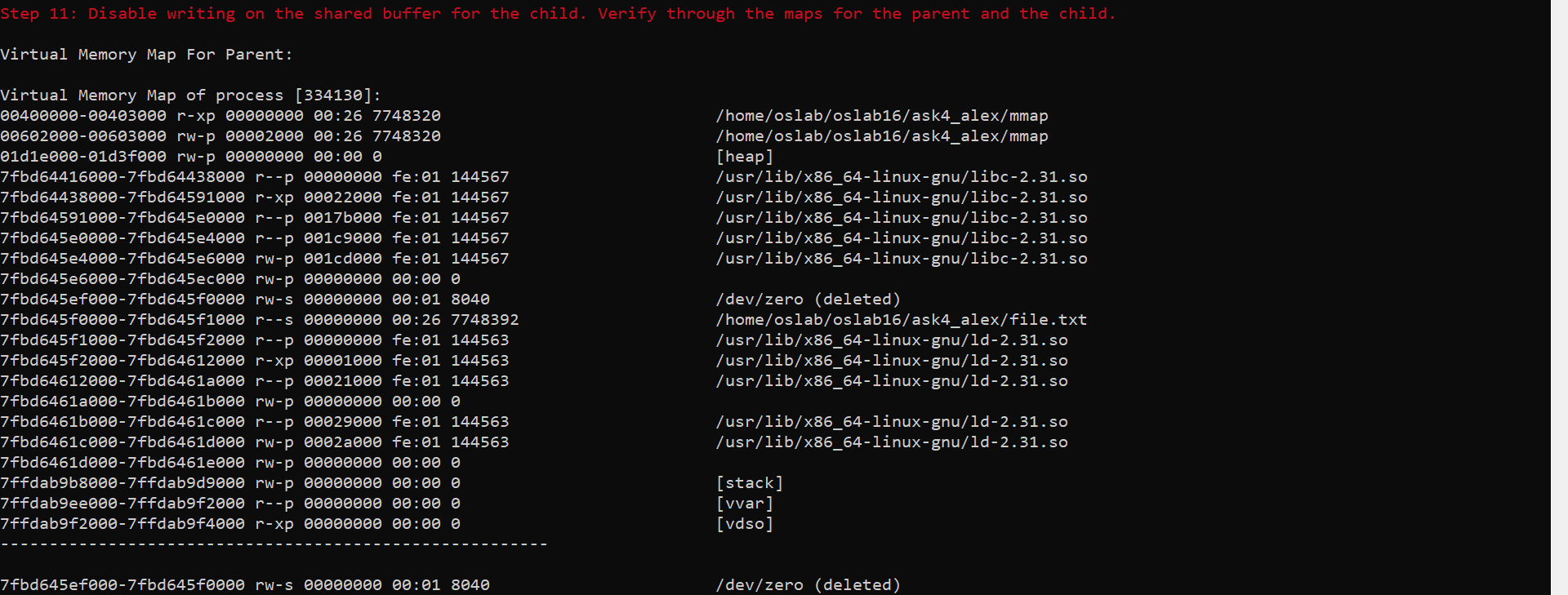


**Βήμα 10**:Ακολουθούμε την ίδια διαδικασία για τον shared buffer και βλέπουμε τώρα ότι η απεικόνιση στην φυσική μνήμη είναι ίδια και για την διεργασία παιδί και για την διεργασία πατέρα. Αυτό συμβαίνει λόγω του flag MAP\_SHARED που μετατρέπει την σελίδα σε διαμοιραζόμενη και επιτρέπει στις διεργασίες να επεξεργάζονται την ίδια μνήμη



**Βήμα 11**:Απαγορεύουμε τις εγγραφές για την διεργασία παιδί μέσω της mprotect(). Τυπώνουμε τους χάρτες εικονικής μνήμης και για την διεργασία παιδί και για την διεργασία πατέρα και μέσω της show\_va\_info() διαπιστώνουμε ότι όντως το δικαίωμα εγγραφής έχει αφαιρεθεί από το παιδί.

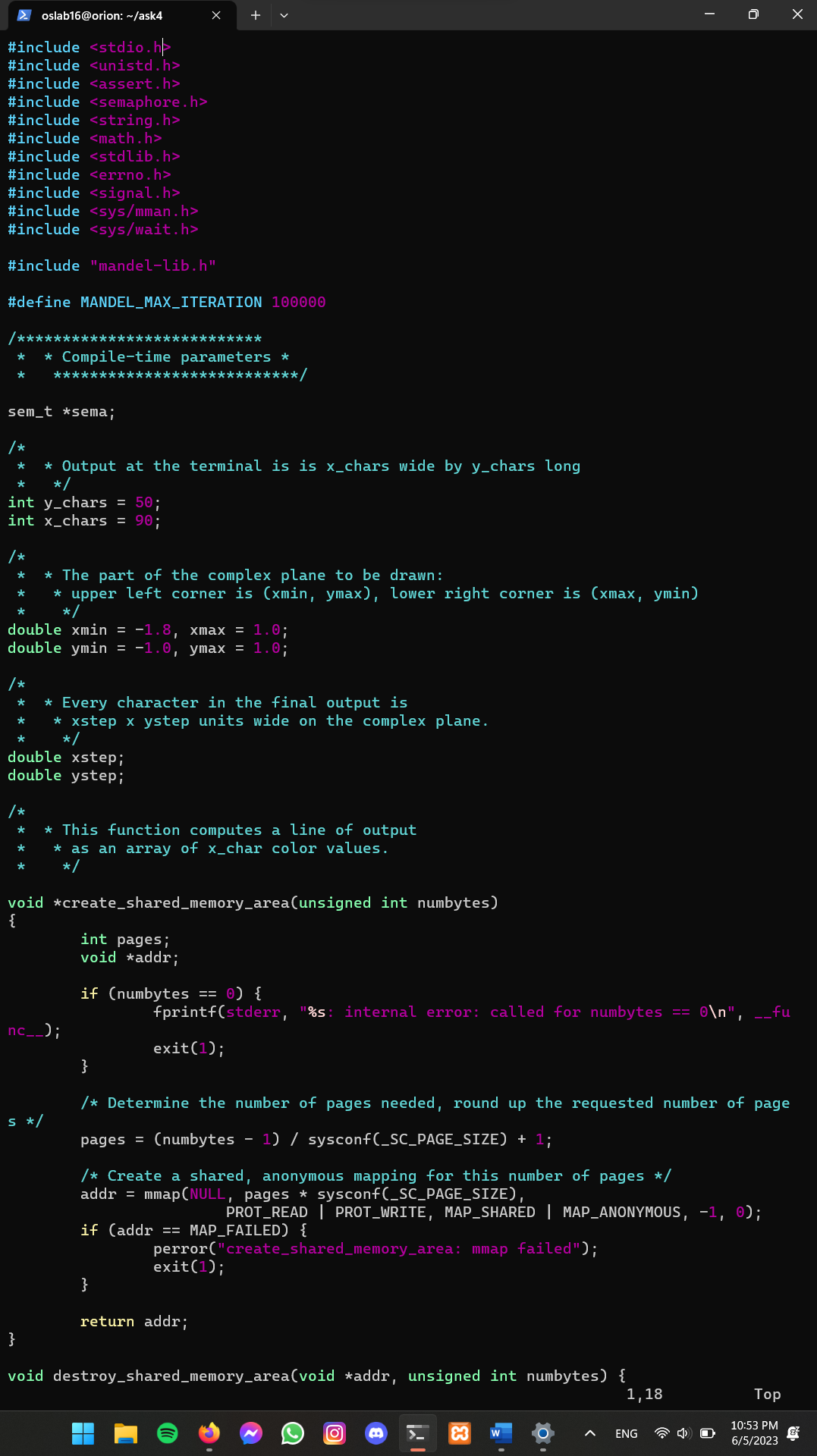


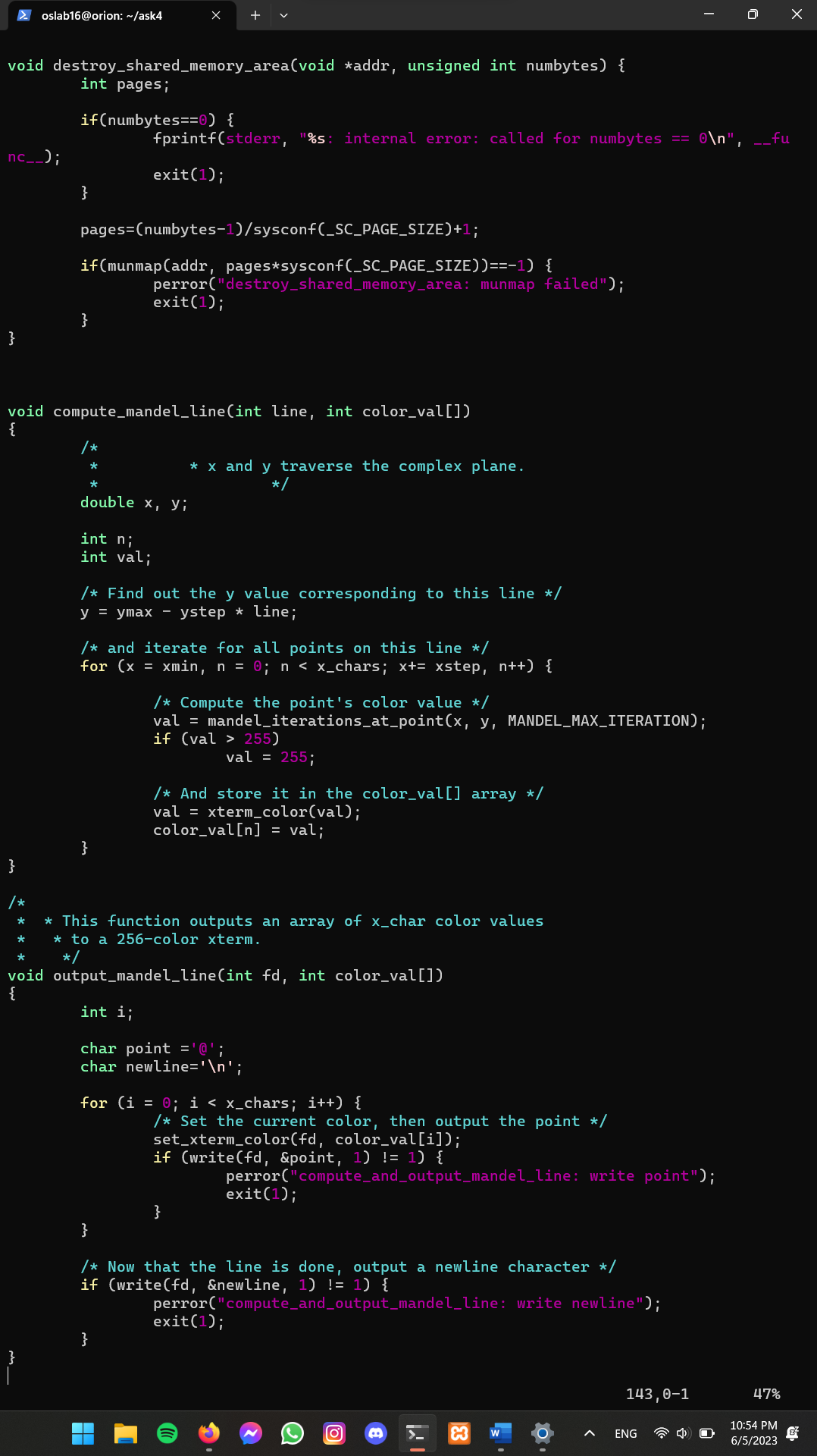


**Βήμα 12**:Αποδεσμεύουμε όλους τους buffers και για τις δύο διεργασίες με χρήση της munmap().

**1.2 Παράλληλος υπολογισμός Mandelbrot με διεργασίες αντί για νήματα**

**1.2.1 Semaphores πάνω από διαμοιραζόμενη μνήμη**





Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

**Ερωτήσεις**

**Ερώτηση 1**

Η υλοποίηση με threads είναι πιο γρήγορη από αυτή των processes, αφού μπορούν να εκμεταλλευτούν καλύτερα την διαμοιραζόμενη μνήμη και να αποφύγουν το overhead της επικοινωνίας των διεργασιών. Επιπλέον, σε multicore συστήματα, η υλοποίηση μπορεί να γίνει αποδοτικότερα παράλληλα μέσω threads.

Η χρήση semaphores για την υλοποίηση της άσκησης προσθέτει παραπάνω overhead, ενώ σε multicore συστήματα, όταν πολλές διεργασίες προσπαθούν να έχουν πρόσβαση στον ίδιο, κάποιες θα χρειαστεί να περιμένουν να ελευθερωθεί, αυξάνοντας ακόμη τον συνολικό χρόνο εκτέλεσης. Επιπλέον αυτή η «επικοινωνία» μεταξύ των διεργασίων, ποια δηλαδή θα έχει πρόσβαση μία δεδομένη χρονική στιγμή στον σημαφόρο, αυξάνει τον χρόνο εκτέλεσης. Τέλος τόσο η δημιουργία όσο και η εναλλαγή μεταξύ των διεργασιών είναι πολύ πιο χρονοβόρα σε σχέση με τα threads.

**1.2.2 Υλοποίηση χωρίς semaphores**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λειτουργικό σύστημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λειτουργικό σύστημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

**Ερωτήσεις**

**Ερώτηση 1**

Στην παραπάνω υλοποίηση χωρίς semaphores ο συγχρονισμός επιτυγχάνεται μέσω ενός 2-διάστατου buffer τον οποίον ορίζουμε ως global στην αρχή του προγράμματός μας. Σε αυτόν τον buffer αναλαμβάνει να τοποθετήσει η κάθε διεργασία παιδί τις γραμμές του Mandelbrot που της αναλογούν αφού πρώτα τις υπολογίσει. Η διεργασία πατέρας αναμένει τις διεργασίες παιδιά να ολοκληρωθούν , δηλαδή να υπολογισθούν όλες οι γραμμές και να τοποθετηθούν στον buffer, και έπειτα τυπώνει τον πίνακα των δεδομένων παράγοντας το σύνολο Mandelbrot.

Εάν ο πίνακας είχε μέγεθος x\_chars x NPROCS αντί για x\_chars x y\_chars τότε δεν θα μπορούσαμε να τυπώσουμε όλο το σύνολο Mandelbrot με την μια αλλά ο υπολογισμός θα έπρεπε να γίνει σε δόσεις. Ένας τρόπος λύσης θα ήταν να χρησιμοποιούσαμε σήματα δηλαδή οι διεργασίες παιδιά αφού υπολογίσουν την μία δόση του Mandelbrot να λαμβάνουν σήμα να περιμένουν μέχρι να τυπωθεί από τον πατέρα και στην συνέχεια να συνεχίζουν με την επόμενη δόση.