ΕΘΝΙΚΟ ΜΕΤΣΟΒΕΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΝΑΦΟΡΑ 3ης ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΣΚΗΣΗΣ

Στοιχεία Ομάδας

- Αναγνωριστικό: oslab005
- Μέλος 1ο: Πέππας Μιχαήλ – Αθανάσιος, Α.Μ: 03121026
- Μέλος 2ο: Σαουνάτσος Ανδρέας, Α.Μ: 03121197
- Ημερομηνία Παράδοσης Αναφοράς: 17.05.2024

- **Ενότητα 1 – Κλήσεις συστήματος και βασικοί μηχανισμοί του ΛΣ για τη διαχείριση της εικονικής μνήμης (Virtual Memory – VM)**

Αρχικά, παραθέτουμε τον συμπληρωμένο κώδικα της άσκησης – του αρχείου mmap.c – ακολούθως, ώστε να μπορεί να γίνει εύκολη αναφορά κατά την παράθεση των απαντήσεων στα ζητούμενα ερωτήματα. Ωστόσο, για λόγους ευκολίας, τα καίρια σημεία του κώδικα θα παρατίθενται εκ νέου σε κάθε υποερώτημα, όπως και η αντίστοιχη έξοδος. Αξίζει να επισημάνουμε ότι κάθε κώδικας που συμπληρώνουμε στο αρχείο φροντίζει να ελέγχει και να προλαμβάνει πιθανά λάθη, όπως ήδη έχουμε μάθει έως τώρα, κάτι στο οποίο (θα φαίνεται) δεν θα αναφερθούμε ξανά. Συνεπώς, ο πηγαίος κώδικας του mmap.c είναι ο κάτωθι:

1. Στην παρακάτω εικόνα, βλέπουμε τον συμπληρωμένο κώδικα για το ζητούμενο ερώτημα:

```
press_enter();
// TODO: Write your code here to complete Step 1.
show_maps();
```

Έτσι, μέσω της κλήσης συστήματος show_maps(), λαμβάνουμε την ακόλουθη έξοδο, δηλαδή τον χάρτη εικονικής μνήμης της τρέχουσας διεργασίας:

```
Step 1: Print the virtual address space map of this process [801788].


Virtual Memory Map of process [801788]:
562ab6011000-562ab6012000 r--p 00000000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6012000-562ab6013000 r-xp 00001000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6013000-562ab6014000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6014000-562ab6015000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6015000-562ab6016000 rw-p 00003000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6052000-562ab6073000 rw-p 00000000 00:00 0                          [heap]
7f4da6897000-7f4da68b9000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da68b9000-7f4da6a12000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a12000-7f4da6a61000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a61000-7f4da6a65000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a65000-7f4da6a67000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a67000-7f4da6a6d000 rw-p 00000000 00:00 0
7f4da6a72000-7f4da6a73000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a73000-7f4da6a93000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a93000-7f4da6a9b000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9c000-7f4da6a9d000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9d000-7f4da6a9e000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9e000-7f4da6a9f000 rw-p 00000000 00:00 0
7ffc666f0000-7ffc66711000 rw-p 00000000 00:00 0                          [stack]
7ffc667df000-7ffc667e3000 r--p 00000000 00:00 0                          [vvar]
7ffc667e3000-7ffc667e5000 r-xp 00000000 00:00 0                          [vdso]
-----------------------------------------------------
```

2. Στην παρακάτω εικόνα, βλέπουμε τον συμπληρωμένο κώδικα για το ζητούμενο ερώτημα:

```
// TODO: Write your code here to complete Step 2.
heap_private_buf = mmap(NULL, buffer_size, PROT_READ|PROT_WRITE, MAP_ANONYMOUS|MAP_PRIVATE, fd, 0);
if(heap_private_buf == MAP_FAILED) {perror("mmap"); return 1;}
show_maps();
```

Έτσι, μέσω της κλήσης συστήματος mmap() με τις κατάλληλες παραμέτρους, λαμβάνουμε – με επιπλέον χρήση της εντολής show_maps() – τον νέο χάρτη εικονικών διευθύνσεων μνήμης, όπως φαίνεται ακολούθως:

```
Step 2: Use mmap(2) to allocate a private buffer of size equal to 1 page and print the VM map again.


Virtual Memory Map of process [801788]:
562ab6011000-562ab6012000 r--p 00000000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6012000-562ab6013000 r-xp 00001000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6013000-562ab6014000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6014000-562ab6015000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6015000-562ab6016000 rw-p 00003000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6052000-562ab6073000 rw-p 00000000 00:00 0                          [heap]
7f4da6897000-7f4da68b9000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da68b9000-7f4da6a12000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a12000-7f4da6a61000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a61000-7f4da6a65000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a65000-7f4da6a67000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a67000-7f4da6a6d000 rw-p 00000000 00:00 0
7f4da6a72000-7f4da6a73000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a73000-7f4da6a93000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a93000-7f4da6a9b000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9b000-7f4da6a9c000 rw-p 00000000 00:00 0
7f4da6a9c000-7f4da6a9d000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9d000-7f4da6a9e000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9e000-7f4da6a9f000 rw-p 00000000 00:00 0
7ffc666f0000-7ffc66711000 rw-p 00000000 00:00 0                          [stack]
7ffc667df000-7ffc667e3000 r--p 00000000 00:00 0                          [vvar]
7ffc667e3000-7ffc667e5000 r-xp 00000000 00:00 0                          [vdso]
------------------------------------------------------
```

3. Κατόπιν, βρίσκουμε και τυπώνουμε τη φυσική μνήμη που δεσμεύσαμε προηγουμένως, μέσω της εντολής get_physical_address(), όπως φαίνεται ακολούθως στον κώδικα:

```
// TODO: Write your code here to complete Step 3.
//get_physical_address((uint64_t)heap_private_buf);
printf("The physical address of buff is: %ld\n",get_physical_address((uint64_t)heap_private_buf));
```

και λαμβάνουμε το ακόλουθο μήνυμα:

```
Step 3: Find and print the physical address of the buffer in main memory. What do you see?

VA[0x7f4da6a9b000] is not mapped; no physical memory allocated.
The physical address of buff is: 0
```

Παρατηρούμε ότι το ΛΣ δεν δέσμευσε φυσική μνήμη από το μηχάνημά μας που να αντιστοιχεί στην παραπάνω εικονική! Η εξήγηση για αυτό είναι ότι μπορεί μεν να εκφράσαμε την ανάγκη μας (δυναμικά) για παραχώρηση επιπλέον μνήμης, ωστόσο αυτή δεν τη χρειαστήκαμε ποτέ και το ΛΣ μας την παραχωρεί μόνο κατά τον χρόνο που τη χρησιμοποιούμε, on demand.

4. Έπειτα, χρησιμοποιούμε την εντολή memset() προκειμένου να γεμίσουμε με μηδενικά τον buffer μας, όπως φαίνεται ακολούθως:

```
// TODO: Write your code here to complete Step 4.
memset(heap_private_buf,0,buffer_size);
printf("The physical address of buff is: %ld\n",get_physical_address((uint64_t)heap_private_buf));
```

και, ομοίως με τη διαδικασία που ακολουθήσαμε προηγουμένως, λαμβάνουμε την ακόλουθη έξοδο:

```
Step 4: Initialize your buffer with zeros and repeat Step 3. What happened?

The physical address of buff is: 9147281408
```

Παρατηρούμε ότι τώρα – που πλέον χρειαστήκαμε και αξιοποιήσαμε ενεργά τη μνήμη που ζητήσαμε από το ΛΣ – αυτό μας την παραχώρησε και έτσι η εικονική διεύθυνση πλέον αντιστοιχεί και σε φυσική: αυτή που εμφανίζεται παραπάνω.

5. Έπειτα, όπως φαίνεται και στο ακόλουθο κομμάτι κώδικα, ανοίγουμε το αρχείο file.txt και βρίσκουμε το μέγεθός του και δεσμεύουμε αντίστοιχο μέγεθος μνήμης, μέσω της mmap():

```
//TODO: Write your code here to complete Step 5.
fd=open("file.txt",O_RDONLY);
if(fd == -1) die("open");

struct stat st;
size_t file_size;
if (stat("file.txt", &st) == 0)
        file_size = st.st_size;
else {perror("stat"); exit(1);}

file_shared_buf=mmap(NULL,file_size,PROT_READ,MAP_SHARED,fd,0);
if(file_shared_buf == MAP_FAILED) die("mmap");

write(0,file_shared_buf,file_size);
close(fd);

show_maps();
show_va_info((uint64_t)file_shared_buf);
```

Παρατηρούμε ότι στην έξοδό μας βρίσκεται τόσο το περιεχόμενο του αρχείου (“Hello everyone”) όσο και ο νέος πίνακα εικονικής μνήμης της διεργασίας, όπως και η θέση του αρχείου file.txt σε αυτή. Για να την απομονώσουμε, χρησιμοποιήσαμε την εντολή show_va_info() της μνήμης που δεσμεύσαμε:

```
Step 5: Use mmap(2) to read and print file.txt. Print the new mapping information that has been created.

Hello everyone!

Virtual Memory Map of process [801788]:
562ab6011000-562ab6012000 r--p 00000000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6012000-562ab6013000 r-xp 00001000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6013000-562ab6014000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6014000-562ab6015000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6015000-562ab6016000 rw-p 00003000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6052000-562ab6073000 rw-p 00000000 00:00 0                          [heap]
7f4da6897000-7f4da68b9000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da68b9000-7f4da6a12000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a12000-7f4da6a61000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a61000-7f4da6a65000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a65000-7f4da6a67000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a67000-7f4da6a6d000 rw-p 00000000 00:00 0
7f4da6a71000-7f4da6a72000 r--s 00000000 00:26 2246818                    /home/oslab/oslab005/askhsh3/file.txt
7f4da6a72000-7f4da6a73000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a73000-7f4da6a93000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a93000-7f4da6a9b000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9b000-7f4da6a9c000 rw-p 00000000 00:00 0
7f4da6a9c000-7f4da6a9d000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9d000-7f4da6a9e000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9e000-7f4da6a9f000 rw-p 00000000 00:00 0
7ffc666f0000-7ffc66711000 rw-p 00000000 00:00 0                          [stack]
7ffc667df000-7ffc667e3000 r--p 00000000 00:00 0                          [vvar]
7ffc667e3000-7ffc667e5000 r-xp 00000000 00:00 0                          [vdso]
-----------------------------------------------------------

7f4da6a71000-7f4da6a72000 r--s 00000000 00:26 2246818                    /home/oslab/oslab005/askhsh3/file.txt
```

6. Τέλος, πριν εκτελέσουμε τη fork(), συντάσσουμε τον ακόλουθο κώδικα, στον οποίο χρησιμοποιούμε τη memset() για να αρχικοποιήσουμε την εικονική μνήμη που δεσμεύσαμε μέσω της mmap(), ως εξής:

```
//TODO: Write your code here to complete Step 6.
fd=-1;
heap_shared_buf = mmap(NULL, buffer_size, PROT_READ|PROT_WRITE, MAP_ANONYMOUS|MAP_SHARED, fd, 0);
if(heap_shared_buf==MAP_FAILED) die("mmap");
memset(heap_shared_buf,0,buffer_size);
printf("The physical address of the new shared memory is: %ld\n",get_physical_address((uint64_t)heap_shared_buf));
show_maps();
show_va_info((uint64_t)heap_shared_buf);
```

και κατόπιν απεικονίζουμε τις νέες πληροφορίες, όπως και στα προηγούμενα ερωτήματα:

```
Step 6: Use mmap(2) to allocate a shared buffer of size equal to 1 page. Initialize the buffer and print the new mapping information that has been created.

The physical address of the new shared memory is: 7833841664

Virtual Memory Map of process [801788]:
562ab6011000-562ab6012000 r--p 00000000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6012000-562ab6013000 r-xp 00001000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6013000-562ab6014000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6014000-562ab6015000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6015000-562ab6016000 rw-p 00003000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6052000-562ab6073000 rw-p 00000000 00:00 0                          [heap]
7f4da6897000-7f4da68b9000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da68b9000-7f4da6a12000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a12000-7f4da6a61000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a61000-7f4da6a65000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a65000-7f4da6a67000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a67000-7f4da6a6d000 rw-p 00000000 00:00 0
7f4da6a70000-7f4da6a71000 rw-s 00000000 00:01 554                        /dev/zero (deleted)
7f4da6a71000-7f4da6a72000 r--s 00000000 00:26 2246818                    /home/oslab/oslab005/askhsh3/file.txt
7f4da6a72000-7f4da6a73000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a73000-7f4da6a93000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a93000-7f4da6a9b000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9b000-7f4da6a9c000 rw-p 00000000 00:00 0
7f4da6a9c000-7f4da6a9d000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9d000-7f4da6a9e000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9e000-7f4da6a9f000 rw-p 00000000 00:00 0
7ffc666f0000-7ffc66711000 rw-p 00000000 00:00 0                          [stack]
7ffc667df000-7ffc667e3000 r--p 00000000 00:00 0                          [vvar]
7ffc667e3000-7ffc667e5000 r-xp 00000000 00:00 0                          [vdso]
-----------------------------------------------------

7f4da6a70000-7f4da6a71000 rw-s 00000000 00:01 554                        /dev/zero (deleted)
```

Στο σημείο αυτό, εκτελούμε fork() και δημιουργούμε μια νέα διεργασία.

7. Αρχικά, συμπληρώνουμε τον κώδικα του ερωτήματος για τον πατέρα και το παιδί, ώστε να τυπώνουν τον χάρτη μνήμης ως εξής:

```
//TODO: Write your code here to complete child's part of Step 7.
printf("The memory map of the child is:\n");
show_maps();
```

```
//TODO: Write your code here to complete parent's part of Step 7.
printf("The memory map of the parent is:\n");
show_maps();
```

και το αποτέλεσμα φαίνεται ακολούθως:

```
Step 7: Print parent's and child's map.

The memory map of the parent is:

Virtual Memory Map of process [801788]:
562ab6011000-562ab6012000 r--p 00000000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6012000-562ab6013000 r-xp 00001000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6013000-562ab6014000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6014000-562ab6015000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6015000-562ab6016000 rw-p 00003000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6052000-562ab6073000 rw-p 00000000 00:00 0                          [heap]
7f4da6897000-7f4da68b9000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da68b9000-7f4da6a12000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a12000-7f4da6a61000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a61000-7f4da6a65000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a65000-7f4da6a67000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a67000-7f4da6a6d000 rw-p 00000000 00:00 0
7f4da6a70000-7f4da6a71000 rw-s 00000000 00:01 554                        /dev/zero (deleted)
7f4da6a71000-7f4da6a72000 r--s 00000000 00:26 2246818                    /home/oslab/oslab005/askhsh3/file.txt
7f4da6a72000-7f4da6a73000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a73000-7f4da6a93000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a93000-7f4da6a9b000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9b000-7f4da6a9c000 rw-p 00000000 00:00 0
7f4da6a9c000-7f4da6a9d000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9d000-7f4da6a9e000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9e000-7f4da6a9f000 rw-p 00000000 00:00 0
7ffc666f0000-7ffc66711000 rw-p 00000000 00:00 0                          [stack]
7ffc667df000-7ffc667e3000 r--p 00000000 00:00 0                          [vvar]
7ffc667e3000-7ffc667e5000 r-xp 00000000 00:00 0                          [vdso]
-----------------------------------------------------

The memory map of the child is:

Virtual Memory Map of process [801789]:
562ab6011000-562ab6012000 r--p 00000000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6012000-562ab6013000 r-xp 00001000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6013000-562ab6014000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6014000-562ab6015000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6015000-562ab6016000 rw-p 00003000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6052000-562ab6073000 rw-p 00000000 00:00 0                          [heap]
7f4da6897000-7f4da68b9000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da68b9000-7f4da6a12000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a12000-7f4da6a61000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a61000-7f4da6a65000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a65000-7f4da6a67000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a67000-7f4da6a6d000 rw-p 00000000 00:00 0
7f4da6a70000-7f4da6a71000 rw-s 00000000 00:01 554                        /dev/zero (deleted)
7f4da6a71000-7f4da6a72000 r--s 00000000 00:26 2246818                    /home/oslab/oslab005/askhsh3/file.txt
7f4da6a72000-7f4da6a73000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a73000-7f4da6a93000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a93000-7f4da6a9b000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9b000-7f4da6a9c000 rw-p 00000000 00:00 0
7f4da6a9c000-7f4da6a9d000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9d000-7f4da6a9e000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9e000-7f4da6a9f000 rw-p 00000000 00:00 0
7ffc666f0000-7ffc66711000 rw-p 00000000 00:00 0                          [stack]
7ffc667df000-7ffc667e3000 r--p 00000000 00:00 0                          [vvar]
7ffc667e3000-7ffc667e5000 r-xp 00000000 00:00 0                          [vdso]
-----------------------------------------------------
```

Παρατηρούμε ότι ο χάρτης μνήμης που τυπώθηκε είναι ο ίδιος, τόσο για τη διεργασία – γονέα όσο και για τη διεργασία – παιδί, κάτι το οποίο είναι αναμενόμενο, καθώς η fork() δημιουργεί ένα αντίγραφο του γονέα στο παιδί, τη στιγμή που αυτό γεννιέται, με όλα τα στοιχεία του. Το δικαίωμα εγγραφής αφαιρείται τόσο από τον γονέα όσο και από το παιδί, αφού έχουμε Copy-On-Write (COW).

8. Ομοίως, συμπληρώνουμε τον ζητούμενο κώδικα για τις δύο διεργασίες, ώστε να τυπώνει τη φυσική διεύθυνση της εικονικής που δεσμεύσαμε, ως εξής:

```
//TODO: Write your code here to complete child's part of Step 8.
 printf("The physical address of private buff for child is: %ld\n",get_physical_address((uint64_t)heap_private_buf));
```

```
//TODO: Write your code here to complete parent's part of Step 8.
printf("The physical address of private buff for parent is: %ld\n",get_physical_address((uint64_t)heap_private_buf));
```

και το αποτέλεσμα φαίνεται ακολούθως:

```
 Step 8: Find the physical address of the private heap buffer (main) for both the parent and the child.

 The physical address of private buff for parent is: 9147281408
 The physical address of private buff for child is: 9147281408
```

Παρατηρούμε ότι η διεύθυνση φυσικής μνήμης που τυπώθηκε είναι η ίδια, τόσο για τη διεργασία – γονέα όσο και για τη διεργασία – παιδί, κάτι το οποίο είναι αναμενόμενο, καθώς η fork() δημιουργεί ένα αντίγραφο του γονέα στο παιδί, τη στιγμή που αυτό γεννιέται, με όλα τα στοιχεία του.

9. Τώρα, γράφουμε στον private buffer από τη διεργασία παιδί και επαναλαμβάνουμε τη διαδικασία, όπως στο προηγούμενο ερώτημα:

```
//TODO: Write your code here to complete parent's part of Step 9.
printf("The new physical address of private buff for parent is: %ld\n",get_physical_address((uint64_t)heap_private_buf));
```

```
 //TODO: Write your code here to complete child's part of Step 9.
 memset(heap_private_buf,1,buffer_size);
 printf("The new physical address of private buff for child is: %ld\n",get_physical_address((uint64_t)heap_private_buf));
```

και λαμβάνουμε την ακόλουθη έξοδο:

```
 Step 9: Write to the private buffer from the child and repeat step 8. What happened?

 The new physical address of private buff for parent is: 9147281408
 The new physical address of private buff for child is: 5582807040
```

Παρατηρούμε ότι η διεύθυνση φυσικής μνήμης, στην οποία αντιστοιχεί ο heap_private_buf, έχει παραμείνει η ίδια για τη διεργασία – γονέα, ενώ έχει αλλάξει για τη διεργασία παιδί. Αυτό ήταν αναμενόμενο, αφού το παιδί πλέον ζήτησε να γράψει ενεργά στη μνήμη και έτσι το αντίγραφό του έγινε ξεχωριστό από αυτό του πατέρα του, του αποδόθηκε δικαίωμα εγγραφής από το ΛΣ, καθώς και μία φυσική θέση μνήμης, προκειμένου να αντιγραφεί το αρχικό περιεχόμενο και να γίνουν οι απαραίτητες αλλαγές. Δηλαδή, αρχικά το περιεχόμενο του buf δεν είχε αντιγραφεί για τη διεργασία παιδί από το ΛΣ όταν έγινε το fork() (σε φυσική μνήμη, αφού δεν χρειαζόταν να το εγγράψει και παρέμενε σε μια εικονική μνήμη, ενώ τώρα αντιγράφηκε σε φυσική μνήμη, με δικαίωμα εγγραφής. Αυτό συνέβη επειδή η mmap() κλήθηκε με MAP_PRIVATE και έχουμε COW.

10. Τώρα, γράφουμε στον shared buffer από τη διεργασία παιδί και επαναλαμβάνουμε τη διαδικασία, όπως στο προηγούμενο ερώτημα:

```
//TODO: Write your code here to complete parent's part of Step 10.
printf("The new physical address of the parent's shared memory is: %ld\n",get_physical_address((uint64_t)heap_shared_buf));
```

```
//TODO: Write your code here to complete child's part of Step 10.
memset(heap_shared_buf,1,buffer_size);
printf("The new physical address of the child's shared memory is: %ld\n",get_physical_address((uint64_t)heap_shared_buf));
```

και λαμβάνουμε την ακόλουθη έξοδο:

```
Step 10: Write to the shared heap buffer (main) from child and get the physical address for both the parent and the child. What happened?

The new physical address of the parent's shared memory is: 7833841664
The new physical address of the child's shared memory is: 7833841664
```

Παρατηρούμε ότι η διεύθυνση φυσικής μνήμης, στην οποία αντιστοιχεί ο heap_shared_buf, έχει παραμείνει η ίδια για τη διεργασία – γονέα και για τη διεργασία παιδί, ενώ είναι ίδια και με την αρχική! Αυτό ήταν αναμενόμενο, αφού η mmap() κλήθηκε με MAP_SHARED και έχουμε COW, επιτρέποντας τη διεργασιακή επικοινωνία. Δηλαδή, αρχικά το περιεχόμενο του buf δεν είχε αντιγραφεί για τη διεργασία παιδί από το ΛΣ όταν έγινε το fork() (σε φυσική μνήμη), αφού δεν χρειαζόταν να το εγγράψει και παρέμενε σε μια εικονική μνήμη, ενώ τώρα αντιγράφηκε σε φυσική μνήμη, με δικαίωμα εγγραφής.

11. Έπειτα, απαγορεύουμε τις εγγραφές στον shared buffer για τη διεργασία – παιδί, κάνοντας χρήση της εντολής mprotect(), ως εξής:

```
//TODO: Write your code here to complete parent's part of Step 11.
printf("The memory map of the parent is:\n");
show_maps();
show_va_info((uint64_t)heap_shared_buf);
```

```
//TODO: Write your code here to complete child's part of Step 11.
if (mprotect(heap_shared_buf, buffer_size,PROT_READ) == -1)
        perror("mprotect");
printf("The memory map of the child is:\n");
show_maps();
show_va_info((uint64_t)heap_shared_buf);
```

και λαμβάνουμε την ακόλουθη έξοδο:

```
Step 11: Disable writing on the shared buffer for the child. Verify through the maps for the parent and the child.

The memory map of the parent is:

Virtual Memory Map of process [801788]:
562ab6011000-562ab6012000 r--p 00000000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6012000-562ab6013000 r-xp 00001000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6013000-562ab6014000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6014000-562ab6015000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6015000-562ab6016000 rw-p 00003000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6052000-562ab6073000 rw-p 00000000 00:00 0                          [heap]
7f4da6897000-7f4da68b9000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da68b9000-7f4da6a12000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a12000-7f4da6a61000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a61000-7f4da6a65000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a65000-7f4da6a67000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a67000-7f4da6a6d000 rw-p 00000000 00:00 0
7f4da6a70000-7f4da6a71000 rw-s 00000000 00:01 554                        /dev/zero (deleted)
7f4da6a71000-7f4da6a72000 r--s 00000000 00:26 2246818                    /home/oslab/oslab005/askhsh3/file.txt
7f4da6a72000-7f4da6a73000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a73000-7f4da6a93000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a93000-7f4da6a9b000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9b000-7f4da6a9c000 rw-p 00000000 00:00 0
7f4da6a9c000-7f4da6a9d000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9d000-7f4da6a9e000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9e000-7f4da6a9f000 rw-p 00000000 00:00 0
7ffc666f0000-7ffc66711000 rw-p 00000000 00:00 0                          [stack]
7ffc667df000-7ffc667e3000 r--p 00000000 00:00 0                          [vvar]
7ffc667e3000-7ffc667e5000 r-xp 00000000 00:00 0                          [vdso]
-----------------------------------------------------

7f4da6a70000-7f4da6a71000 rw-s 00000000 00:01 554                        /dev/zero (deleted)
The memory map of the child is:

Virtual Memory Map of process [801789]:
562ab6011000-562ab6012000 r--p 00000000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6012000-562ab6013000 r-xp 00001000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6013000-562ab6014000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6014000-562ab6015000 r--p 00002000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6015000-562ab6016000 rw-p 00003000 00:26 2246822                    /home/oslab/oslab005/askhsh3/mmap
562ab6052000-562ab6073000 rw-p 00000000 00:00 0                          [heap]
7f4da6897000-7f4da68b9000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da68b9000-7f4da6a12000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a12000-7f4da6a61000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a61000-7f4da6a65000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a65000-7f4da6a67000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f4da6a67000-7f4da6a6d000 rw-p 00000000 00:00 0
7f4da6a70000-7f4da6a71000 r--s 00000000 00:01 554                        /dev/zero (deleted)
7f4da6a71000-7f4da6a72000 r--s 00000000 00:26 2246818                    /home/oslab/oslab005/askhsh3/file.txt
7f4da6a72000-7f4da6a73000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a73000-7f4da6a93000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a93000-7f4da6a9b000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9b000-7f4da6a9c000 rw-p 00000000 00:00 0
7f4da6a9c000-7f4da6a9d000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9d000-7f4da6a9e000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f4da6a9e000-7f4da6a9f000 rw-p 00000000 00:00 0
7ffc666f0000-7ffc66711000 rw-p 00000000 00:00 0                          [stack]
7ffc667df000-7ffc667e3000 r--p 00000000 00:00 0                          [vvar]
7ffc667e3000-7ffc667e5000 r-xp 00000000 00:00 0                          [vdso]
-----------------------------------------------------

7f4da6a70000-7f4da6a71000 r--s 00000000 00:01 554                        /dev/zero (deleted)
```

Παρατηρούμε ότι η απαγόρευση εγγραφής έχει ολοκληρωθεί για το παιδί και πως αν αυτό πάει να γράψει θα έχουμε segmentation fault.

12. Τέλος, αποδεσμεύουμε όλους τους buffers, κάνοντας χρήση της εντολής munmap(), ως εξής:

```
//TODO: Write your code here to complete parent's part of Step 12.
if(munmap(heap_shared_buf, buffer_size)==-1) perror("munmap");
if(munmap(heap_private_buf, buffer_size)==-1) perror("munmap");
if(munmap(file_shared_buf, file_size)==-1) perror("munmap");
```

```
//TODO: Write your code here to complete child's part of Step 12.
if(munmap(heap_shared_buf, buffer_size)==-1) perror("munmap");
if(munmap(heap_private_buf, buffer_size)==-1) perror("munmap");
if(munmap(file_shared_buf, file_size)==-1) perror("munmap");
```

- **Ενότητα 2 – Παράλληλος υπολογισμός Mandelbrot με διεργασίες αντί για νήματα**

2.1 – Semaphores πάνω από διαμοιραζόμενη μνήμη

Αρχικά, παραθέτουμε τον συμπληρωμένο κώδικα της άσκησης – του αρχείου mandel-fork1.c – ακολούθως, στον οποίο έχουμε υλοποιήσει το mandel με σημαφόρους και διεργασίες, αντί για σημαφόρους και threads. Αξίζει να επισημάνουμε ότι κάθε κώδικας που συμπληρώνουμε στο αρχείο φροντίζει να ελέγχει και να προλαμβάνει πιθανά λάθη, όπως ήδη έχουμε μάθει έως τώρα, κάτι στο οποίο (θα φαίνεται) δεν θα αναφερθούμε ξανά. Συνεπώς, ο πηγαίος κώδικας του mandel-fork1.c είναι ο κάτωθι:

Ερωτήσεις:

1. Ανάμεσα στις υλοποιήσεις με threads και με forks, αναμένουμε καλύτερη επίδοση να έχουμε στην περίπτωση των threads. Αυτό συμβαίνει διότι κατά τη διαδικασία του fork αντιγράφεται το περιεχόμενο της γονικής διεργασίας, μια διαδικασία πολύ χρονοβόρα, αφού πρέπει να σωθεί η κατάσταση της διεργασίας από το ΛΣ στο Process Control Block. Ακόμα, η επικοινωνία μέσω διαμοιραζόμενης μνήμης είναι από μόνη της αργή, καθώς αρχικά αφαιρείται το δικαίωμα εγγραφής από τα παιδιά (έχουμε Copy-On-Write) και από τον πατέρα για τη μνήμη που αυτός έχει δεσμεύσει, ενώ όταν αυτά ζητήσουν να γράψουν τους παραχωρείται εκ νέου. Τέλος, το ίδιο το ΛΣ δημιουργεί και καταστρέφει threads πιο γρήγορα από ό,τι δημιουργεί και καταστρέφει διεργασίες, αφού τα threads έχουν εξαρχής κοινή μνήμη (τις global μεταβλητές) και έτσι δεν χρειάζεται να δεσμεύσουμε εμείς κοινή μνήμη δυναμικά, μέσω της mmap() και της create_shared_memory_area(). Ο παραπάνω ισχυρισμός επιβεβαιώθηκε και με κατάλληλη χρήση της εντολής time για τη χρονομέτρηση του προγράμματός μας, όπου η υλοποίηση με forks ήταν πολύ πιο αργή. Το αποτέλεσμα (χρόνος) με forks (ας γίνει σύγκριση με την προηγούμενη αναφορά) φαίνεται παρακάτω.

Εκτέλεση με 2 διεργασίες:

```
real    0m0.312s
user    0m0.543s
sys     0m0.059s
```

Εκτέλεση με 6 διεργασίες:

```
real    0m0.146s
user    0m0.693s
sys     0m0.018s
```

2. Δυστυχώς, το mmap interface δεν μπορεί να χρησιμοποιηθεί για τον διαμοιρασμό μνήμης μεταξύ διεργασιών που δεν έχουν κοινό ancestor. Αυτό συμβαίνει διότι σε κάθε fork() όλος ο χώρος του πατέρα – επομένως μαζί και ο χώρος εικονικών διευθύνσεων – αντιγράφεται στο παιδί. Συνεπώς, αν ο πατέρας κάνει mmap (shared), τότε η εικονική μνήμη που θα δεσμεύσει θα είναι προσβάσιμη και από το παιδί του, καθώς και από το παιδί του παιδιού του (με την ίδια λογική, αφού το αντίγραφό του θα περιέχει και το αντίγραφο του πατέρα του πατέρα του) και άρα και από οποιονδήποτε απόγονό του, αφού η κατάσταση του αρχικού πατέρα (που περιέχει την κοινή mmap) θα κληροδοτείται σε κάθε επίπεδο. Αυτό, όμως, δεν μπορεί να συμβεί αν δεν έχουμε κοινό πρόγονο, αφού τότε θα κληροδοτηθεί το αντίστοιχο αντίγραφο, το οποίο δεν θα περιέχει την κοινή mmap και άρα ο χώρος αυτός των εικονικών διευθύνσεων δεν θα μπορεί να αξιοποιηθεί.

➢ 2.2 – Υλοποίηση χωρίς semaphores

Αρχικά, παραθέτουμε τον συμπληρωμένο κώδικα της άσκησης – του αρχείου mandel-fork2.c – ακολούθως, στον οποίο έχουμε υλοποιήσει το mandel μόνο με διεργασίες, αντί για σημαφόρους και διεργασίες όπως πριν. Αξίζει να επισημάνουμε ότι κάθε κώδικας που συμπληρώνουμε στο αρχείο φροντίζει να ελέγχει και να προλαμβάνει πιθανά λάθη, όπως ήδη έχουμε μάθει έως τώρα, κάτι στο οποίο (θα φαίνεται) δεν θα αναφερθούμε ξανά. Συνεπώς, ο πηγαίος κώδικας του mandel-fork2.c είναι ο κάτωθι:

Ερωτήσεις

1. Στη συγκεκριμένη υλοποίηση, οι διεργασίες που γεννιούνται και κάνουν το compute της κάθε γραμμής δεν έχουν την ανάγκη για συγχρονισμό, ωστόσο ο γενικότερος συγχρονισμός επιτυγχάνεται μεταξύ όλων των παιδιών και του πατέρα. Αυτό συμβαίνει, διότι αρχικά δεσμεύουμε μία μεγάλη περιοχή μνήμης, μεγέθους (x_chars*y_chars*sizeof(int)), ωστόσο ποτέ καμία διεργασία – παιδί δεν θα πάει στην ίδια «υποπεριοχή» μνήμης της αρχικής μνήμης – μεγέθους x_chars*sizeof(int), που χρειάζεται για κάθε γραμμή. Ειδικότερα, κάθε διεργασία – παιδί έχει τη δική της, αποκλειστική περιοχή – τμήμα μνήμης της αρχικής (διαμοιραζόμενης) που δεσμεύσαμε, με το παραπάνω μέγεθος, στην οποία ποτέ δεν γίνεται πρόσβαση από άλλη διεργασία και όπου αποθηκεύει το compute μιας γραμμής. Έτσι, δεν πρόκειται να «μπλεχτούν» ποτέ τα δεδομένα των computes των γραμμών του mandel και αυτά μπορούν να γίνουν ταυτόχρονα από όλες τις διεργασίες. Όπως και παλιά, το κρίσιμο τμήμα παραμένει αυτό του output της κάθε γραμμής, ωστόσο αυτό το εκτελεί στο τέλος – συνολικά ο πατέρας. Έτσι, ουσιαστικά δεν έχουμε κρίσιμο τμήμα πλέον και ο συγχρονισμός έγκειται στο γεγονός ότι το παλιό κρίσιμο τμήμα έχει απομονωθεί και έχει αποδοθεί πλήρως στον πατέρα, μετά τον τερματισμό όλων των παιδιών του.

   Σε περίπτωση που είχαμε έναν μικρότερο buffer – διαστάσεων nprocs*x_chars – τότε θα ακολουθούσαμε την εξής λογική: καθεμία από τις n διεργασίες θα υπολόγιζε (κατά σειρά) μία από τις n πρώτες γραμμές του buffer και όταν τελείωνε, θα ενημέρωνε τη διεργασία – πατέρα. Όταν όλες τερμάτιζαν (άρα ο buffer έχει γεμίσει), τότε ο πατέρας θα τύπωνε το τμηματικό αποτέλεσμα των n πρώτων γραμμών και έπειτα θα τις ειδοποιούσε, ώστε να προβούν στον υπολογισμό των επόμενων n γραμμών, με την ίδια λογική. Η διαδικασία αυτή θα επαναλαμβανόταν, μέχρι να τελειώσουν όλες οι γραμμές του Mandelbrot, οπότε και το συνολικό πρόγραμμα θα τερματίσει.