3η Εργαστηριακή Αναφορά
Μέλη Ομάδας:
Σοφία Σάββα **ΑΜ:**03121189
Ιωάννης Πολυχρονόπουλος **ΑΜ:**03121089

Άσκηση 1

1.1. Κλήσεις συστήματος και βασικοί μηχανισμοί του ΛΣ για τη διαχείριση της εικονικής μνήμης

Σκοπός αυτής της άσκησης είναι η εξοικείωση με το μηχανισμό της εικονικής μνήμης, η κατανόηση της λειτουργίας του και της σχέσης που έχει με τη φυσική. Τέλος θα τον χρησιμοποιήσουμε για την επίτευξη αποδοτικής διεργασικής επικοινωνίας.

Ερμηνεία αποτελεσμάτων:

***Ερώτημα 1*:**
Τυπώνουμε το χάρτη της εικονικής μνήμης της τρέχουσας διεργασίας χρησιμοποιώντας τη δοσμένη συνάρτηση show_maps().

```
---------------------------------------------------------------------------
    printf(RED "\nStep 1: Print the virtual address space map of this "
        "process [%d].\n" RESET, mypid);
    press_enter();
    show_maps();
---------------------------------------------------------------------------
```

```
Step 1: Print the virtual address space map of this process [1727088].


Virtual Memory Map of process [1727088]:
55f7c703b000-55f7c703c000 r--p 00000000 00:27 4329724                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55f7c703c000-55f7c703d000 r-xp 00001000 00:27 4329724                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55f7c703d000-55f7c703e000 r--p 00002000 00:27 4329724                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55f7c703e000-55f7c703f000 r--p 00002000 00:27 4329724                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55f7c703f000-55f7c7040000 rw-p 00003000 00:27 4329724                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55f7c83f4000-55f7c8415000 rw-p 00000000 00:00 0                          [heap]
7f7194615000-7f7194637000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f7194637000-7f7194790000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f7194790000-7f71947df000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f71947df000-7f71947e3000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f71947e3000-7f71947e5000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f71947e5000-7f71947eb000 rw-p 00000000 00:00 0
7f71947f1000-7f71947f2000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f71947f2000-7f7194812000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f7194812000-7f719481a000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f719481b000-7f719481c000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f719481c000-7f719481d000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f719481d000-7f719481e000 rw-p 00000000 00:00 0
7ffddc661000-7ffddc682000 rw-p 00000000 00:00 0                          [stack]
7ffddc779000-7ffddc77d000 r--p 00000000 00:00 0                          [vvar]
7ffddc77d000-7ffddc77f000 r-xp 00000000 00:00 0                          [vdso]
-----------------------------------------------------------
```

Ερώτημα 2:

Κάνουμε allocate μνήμη μεγέθους μια σελίδας με τη χρήση *mmap()* και τυπώνουμε ξανά τον χάρτη εικονικής μνήμης με τη χρήση της *show_va_info()*.Τα ορίσματα της ***mmap*** καθορίζουν με τη σειρά:

- Την διεύθυνση από την οποία θα ξεκινάει το allocated τμήμα της μνήμης (NULL, δηλαδή δεν καθορίζουμε που θα γίνει το mapping)
- Το μέγεθος του τμήματος (buffer_size=1*get_page_size())
- Το επιθυμητό protection της απεικόνισης εικονικής μνήμης. Πρέπει να ταυτίζεται με το file που θα αντιστοιχηθεί στο τμήμα μνήμης που θα δεσμευτεί και καθορίζει αν θα είναι accessible,executable,readable,writable. (PROT_READ|PROT_WRITE έτσι ώστε να μπορούμε να διαβάσουμε απο αυτό και να γράψουμε σε αυτό)
- Τα flags σύμφωνα με τα οποία καθορίζεται αν οι αλλαγές στην απεικόνιση θα γίνονται ορατές από άλλες διεργασίες που χρησιμοποιούν το ίδιο τμήμα εικονικής μνήμης (MAP_PRIVATE|MAP_ANONYMOUS καθώς ζητείται η δημιουργία private buffer και δεν γίνεται αντιστοίχιση με φάκελο)

```
--------------------------------------------------------------------------------
printf(RED "\nStep 2: Use mmap(2) to allocate a private buffer of "
        "size equal to 1 page and print the VM map again.\n" RESET);
press_enter();
heap_private_buf=mmap(NULL,buffer_size,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYM
OUS,fd,0);
//fd=-1 so MAP_ANONYMOUS flag is chosen
if(heap_private_buf==MAP_FAILED){
       perror("mmap");
        exit(EXIT_FAILURE);
}
show_maps();
show_va_info((uint64_t)heap_private_buf);
--------------------------------------------------------------------------------
```

```
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 [1727088]:
55f7c703b000-55f7c703c000 r--p 00000000 00:27 4329724                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55f7c703c000-55f7c703d000 r-xp 00001000 00:27 4329724                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55f7c703d000-55f7c703e000 r--p 00002000 00:27 4329724                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55f7c703e000-55f7c703f000 r--p 00002000 00:27 4329724                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55f7c703f000-55f7c7040000 rw-p 00003000 00:27 4329724                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55f7c83f4000-55f7c8415000 rw-p 00000000 00:00 0                          [heap]
7f7194615000-7f7194637000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f7194637000-7f7194790000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f7194790000-7f71947df000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f71947df000-7f71947e3000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f71947e3000-7f71947e5000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f71947e5000-7f71947eb000 rw-p 00000000 00:00 0
7f71947f1000-7f71947f2000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f71947f2000-7f7194812000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f7194812000-7f719481a000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f719481a000-7f719481b000 rw-p 00000000 00:00 0
7f719481b000-7f719481c000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f719481c000-7f719481d000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f719481d000-7f719481e000 rw-p 00000000 00:00 0
7ffddc661000-7ffddc682000 rw-p 00000000 00:00 0                          [stack]
7ffddc779000-7ffddc77d000 r--p 00000000 00:00 0                          [vvar]
7ffddc77d000-7ffddc77f000 r-xp 00000000 00:00 0                          [vdso]
--------------------------------------------------------

7f719481a000-7f719481b000 rw-p 00000000 00:00 0
```

Ερώτημα 3:

Χρησιμοποιώντας τη δοσμένη συνάρτηση *get_physical_address* βρίσκουμε και τυπώνουμε την φυσική και εικονική διεύθυνση μνήμης του buffer. Παρατηρούμε ότι ενώ έχει γίνει η δέσμευση της εικονικής μνήμης από την mmap δεν έχει γίνει η αντιστοίχιση με τη φυσική μνήμη. Αυτό συμβαίνει διότι τα Linux εφαρμόζουν demand paging που είναι μια τεχνική διαχείρισης μνήμης σύμφωνα με την οποία η αντιστοίχιση της εικονικής και της φυσικής μνήμης δεν γίνεται στην αρχή του προγράμματος αλλά όταν επιχειρήσει το πρόγραμμα να χρησιμοποιήσει τη δεσμευμένη μνήμη. Για αυτό το λόγο δεν γίνεται η αντιστοίχιση στη δέσμευση της μνήμης (δηλαδή με τη χρήση του mmap) και παίρνουμε τα παρακάτω αποτελέσματα.

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

VA[0x7f719481a000] is not mapped; no physical memory allocated.
The physical address of the first page is 0x0
```

```
--------------------------------------------------------------------------------
printf(RED "\nStep 3: Find and print the physical address of the "
       "buffer in main memory. What do you see?\n" RESET);
press_enter();
pa = get_physical_address((uint64_t)heap_private_buf);
printf("The physical address of the first page is 0x%lx\n",pa);
--------------------------------------------------------------------------------
```

Ερώτημα 4:

Αρχικοποιούμε τον buffer με μηδενικά χρησιμοποιώντας τη memset και επαναλαμβάνουμε το προηγούμενο βήμα. Παρατηρούμε ότι εφόσον τώρα έχει γίνει η προσπέλαση του buffer το λειτουργικό σύστημα έχει κάνει την αντιστοίχιση εικονικής και φυσικής μνήμης.

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

The physical address of the first page is 0x165b31000
```

```
--------------------------------------------------------------------------------
printf(RED "\nStep 4: Initialize your buffer with zeros and repeat "
       "Step 3. What happened?\n" RESET);
press_enter();
memset(heap_private_buf,0,buffer_size);
//memset fills first n(3rd arg) bytes of memory area
//pointed to by s(1st arg) with the constant byte c(2nd arg)
pa=get_physical_address((uint64_t)heap_private_buf);
printf("The physical address of the first page is 0x%lx\n ",pa);
--------------------------------------------------------------------------------
```

Ερώτημα 5:

Χρησιμοποιούμε ***mmap()*** και απεικονίζουμε το αρχείο file.txt στο χώρο διευθύνσεων της διεργασίας. Ύστερα τυπώνουμε το περιεχόμενό του. Για να γίνει σωστά η χρήση της mmap πρώτα καλούμε την ***open*** για το file.txt η οποία θα επιστρέψει τον file descriptor *fd* που θα αποτελέσει ένα από τα ορίσματα της πρώτης συνάρτησης. Για να βρεθεί το μέγεθος του αρχείου και κατ' επέκταση το μέγεθος του χώρου που θα δεσμευτεί χρησιμοποιείται το struct τύπου stat και η συνάρτηση fstat η οποία διευκρινίζει ότι το struct sb θα περιέχει πληροφορίες για τον file descriptor που αποτελεί το πρώτο της όρισμα. Καλούμε επομένως την fstat, ελέγχουμε για πιθανά λάθη και αφού δημιουργηθεί η δομή πληροφοριών sb τη χρησιμοποιούμε για να βρούμε το μέγεθος του αρχείου. Έτσι με την κλήση της mmap δεν θα δεσμευτεί απλά χώρος εικονικής μνήμης προκαθορισμένου μεγέθους αλλά αυτός ο χώρος θα αντιστοιχηθεί και στο fd, το file descriptor του αρχείου file.txt.

```
--------------------------------------------------------------------------------
fd=open("file.txt",O_RDONLY);
if(fd==-1){
        perror("open");
        exit(EXIT_FAILURE);}
struct stat sb;
if(fstat(fd,&sb)==-1){
        perror("fstat");
        close(fd);
        exit(EXIT_FAILURE);
    }
file_shared_buf=mmap(NULL,sb.st_size,PROT_READ,MAP_SHARED,fd,0);
if(file_shared_buf==MAP_FAILED){
        perror("mmap");
        close(fd);
        exit(EXIT_FAILURE);}
write(STDOUT_FILENO,file_shared_buf,sb.st_size);//STDOUT_FILENO fd of stdout in <unistd.h>
show_maps();
show_va_info((uint64_t)file_shared_buf);
--------------------------------------------------------------------------------
```

```
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 [1573274]:
55a7dfecf000-55a7dfed0000 r--p 00000000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed0000-55a7dfed1000 r-xp 00001000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed1000-55a7dfed2000 r--p 00002000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed2000-55a7dfed3000 r--p 00002000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed3000-55a7dfed4000 rw-p 00003000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7e18c2000-55a7e18e3000 rw-p 00000000 00:00 0                          [heap]
7f2cb0cc9000-7f2cb0ceb000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0ceb000-7f2cb0e44000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e44000-7f2cb0e93000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e93000-7f2cb0e97000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e97000-7f2cb0e99000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e99000-7f2cb0e9f000 rw-p 00000000 00:00 0
7f2cb0ea3000-7f2cb0ea4000 r--s 00000000 00:26 4330062                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/file.txt
7f2cb0ea4000-7f2cb0ea5000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ea5000-7f2cb0ec5000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ec5000-7f2cb0ecd000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ecd000-7f2cb0ece000 rw-p 00000000 00:00 0
7f2cb0ece000-7f2cb0ecf000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ecf000-7f2cb0ed0000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ed0000-7f2cb0ed1000 rw-p 00000000 00:00 0
7ffdce700000-7ffdce721000 rw-p 00000000 00:00 0                          [stack]
7ffdce7df000-7ffdce7e3000 r--p 00000000 00:00 0                          [vvar]
7ffdce7e3000-7ffdce7e5000 r-xp 00000000 00:00 0                          [vdso]
```

```
7f2cb0ea3000-7f2cb0ea4000 r--s 00000000 00:26 4330062                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/file.txt
```

Ερώτημα 6:

Δεσμεύουμε νέο ,shared, buffer μεγέθους μιας σελίδας και τον αρχικοποιούμε με άσσους με χρήση της memset(). Τυπώνουμε ξανά τον χάρτη.

```
-------------------------------------------------------------------------
heap_shared_buf=mmap(NULL,buffer_size,PROT_READ|PROT_WRITE,MAP_SHARED|MAP_ANONYMOUS,-1,0);
if(heap_shared_buf==MAP_FAILED){
        perror("mmap");
        exit(EXIT_FAILURE);
}
memset(heap_shared_buf,0,buffer_size);//initialization of buffer
show_maps();
show_va_info((uint64_t)heap_shared_buf);
-------------------------------------------------------------------------
```

```
Virtual Memory Map of process [1573274]:
55a7dfecf000-55a7dfed0000 r--p 00000000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed0000-55a7dfed1000 r-xp 00001000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed1000-55a7dfed2000 r--p 00002000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed2000-55a7dfed3000 r--p 00002000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed3000-55a7dfed4000 rw-p 00003000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7e18c2000-55a7e18e3000 rw-p 00000000 00:00 0                          [heap]
7f2cb0cc9000-7f2cb0ceb000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0ceb000-7f2cb0e44000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e44000-7f2cb0e93000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e93000-7f2cb0e97000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e97000-7f2cb0e99000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e99000-7f2cb0e9f000 rw-p 00000000 00:00 0
7f2cb0ea2000-7f2cb0ea3000 rw-s 00000000 00:01 12232                      /dev/zero (deleted)
7f2cb0ea3000-7f2cb0ea4000 r--s 00000000 00:26 4330062                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/file.txt
7f2cb0ea4000-7f2cb0ea5000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ea5000-7f2cb0ec5000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ec5000-7f2cb0ecd000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ecd000-7f2cb0ece000 rw-p 00000000 00:00 0
7f2cb0ece000-7f2cb0ecf000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ecf000-7f2cb0ed0000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ed0000-7f2cb0ed1000 rw-p 00000000 00:00 0
7ffdce700000-7ffdce721000 rw-p 00000000 00:00 0                          [stack]
7ffdce7df000-7ffdce7e3000 r--p 00000000 00:00 0                          [vvar]
7ffdce7e3000-7ffdce7e5000 r-xp 00000000 00:00 0                          [vdso]
-------------------------------------------------------

7f2cb0ea2000-7f2cb0ea3000 rw-s 00000000 00:01 12232                      /dev/zero (deleted)
```

Ερώτημα 7:

Καλούμε fork [`p = fork();`]και δημιουργείται η διεργασία παιδί. Τυπώνουμε τον χάρτη μνήμης της διεργασίας πατέρα και της διεργασίας παιδιού (χρήση `show_maps()` και στις δύο διεργασίες). Παρατηρούμε ότι καθώς η νέα διεργασία δημιουργείται μέσω της κλήσης της fork() ο χάρτης εικονικής μνήμης της είναι πανομοιότυπος με της parent διεργασίας. Αυτό συμβαίνει διότι όταν καλείται fork η νέα διεργασία που δημιουργείται κληρονομεί ένα αντίγραφο της εικονικής μνήμης, των open file descriptors και του πίνακα σελίδων της parent. Στον πίνακα σελίδων έχουν αφαιρεθεί τα write δικαιώματα και από τις δύο διεργασίες όμως στον χάρτη μνήμης έχουν παραμείνει. Έτσι όταν γίνει απόπειρα εγγραφής απο μια από τις δύο διεργασίες θα γίνει page fault , θα επιβεβαιώσουμε απο τον χάρτη μνήμης ότι η διεργασία έχει permission εγγραφής και θα γίνει αντιγραφή του κοινού τμήματος μνήμης σε μια άλλη φυσική θεση.

```
-------------------------------------------------------------------------
// Child
printf("Virtual Memory of child process:\n");
show_maps();
// Parent
printf("Virtual Memory of parent process: \n");
show_maps();
-------------------------------------------------------------------------
```

Parent Process:

```
Virtual Memory of parent process:

Virtual Memory Map of process [1573274]:
55a7dfecf000-55a7dfed0000 r--p 00000000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed0000-55a7dfed1000 r-xp 00001000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed1000-55a7dfed2000 r--p 00002000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed2000-55a7dfed3000 r--p 00002000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed3000-55a7dfed4000 rw-p 00003000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7e18c2000-55a7e18e3000 rw-p 00000000 00:00 0                          [heap]
7f2cb0cc9000-7f2cb0ceb000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0ceb000-7f2cb0e44000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e44000-7f2cb0e93000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e93000-7f2cb0e97000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e97000-7f2cb0e99000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e99000-7f2cb0e9f000 rw-p 00000000 00:00 0
7f2cb0ea2000-7f2cb0ea3000 rw-s 00000000 00:01 12232                      /dev/zero (deleted)
7f2cb0ea3000-7f2cb0ea4000 r--s 00000000 00:26 4330062                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/file.txt
7f2cb0ea4000-7f2cb0ea5000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ea5000-7f2cb0ec5000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ec5000-7f2cb0ecd000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ecd000-7f2cb0ece000 rw-p 00000000 00:00 0
7f2cb0ece000-7f2cb0ecf000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ecf000-7f2cb0ed0000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ed0000-7f2cb0ed1000 rw-p 00000000 00:00 0
7ffdce700000-7ffdce721000 rw-p 00000000 00:00 0                          [stack]
7ffdce7df000-7ffdce7e3000 r--p 00000000 00:00 0                          [vvar]
7ffdce7e3000-7ffdce7e5000 r-xp 00000000 00:00 0                          [vdso]
```

Child Process:

```
Virtual Memory of child process:

Virtual Memory Map of process [1573275]:
55a7dfecf000-55a7dfed0000 r--p 00000000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed0000-55a7dfed1000 r-xp 00001000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed1000-55a7dfed2000 r--p 00002000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed2000-55a7dfed3000 r--p 00002000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed3000-55a7dfed4000 rw-p 00003000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7e18c2000-55a7e18e3000 rw-p 00000000 00:00 0                          [heap]
7f2cb0cc9000-7f2cb0ceb000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0ceb000-7f2cb0e44000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e44000-7f2cb0e93000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e93000-7f2cb0e97000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e97000-7f2cb0e99000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e99000-7f2cb0e9f000 rw-p 00000000 00:00 0
7f2cb0ea2000-7f2cb0ea3000 rw-s 00000000 00:01 12232                      /dev/zero (deleted)
7f2cb0ea3000-7f2cb0ea4000 r--s 00000000 00:26 4330062                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/file.txt
7f2cb0ea4000-7f2cb0ea5000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ea5000-7f2cb0ec5000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ec5000-7f2cb0ecd000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ecd000-7f2cb0ece000 rw-p 00000000 00:00 0
7f2cb0ece000-7f2cb0ecf000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ecf000-7f2cb0ed0000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ed0000-7f2cb0ed1000 rw-p 00000000 00:00 0
7ffdce700000-7ffdce721000 rw-p 00000000 00:00 0                          [stack]
7ffdce7df000-7ffdce7e3000 r--p 00000000 00:00 0                          [vvar]
7ffdce7e3000-7ffdce7e5000 r-xp 00000000 00:00 0                          [vdso]
------------------------------------------------------------
```

Ερώτηση 8:

Βρίσκουμε και τυπώνουμε τη φυσική διεύθυνση του private buffer για τις διεργασίες parent και child. Παρατηρούμε ότι αρχικά ο private buffer έχει όχι μόνο κοινή virtual address αλλά και physical address στις δύο διεργασίες. Αυτό συμβαίνει λόγω των κοινών memory pages που έχουν οι διεργασίες αρχικά. Η κοινή χρήση των φυσικών σελίδων μνήμης μεταξύ του parent και του child συμβαίνει για να εξοικονομηθούν πόροι και να αυξηθεί η απόδοση του συστήματος ενώ φυσικές διευθύνσεις μνήμης παραμένουν κοινές μέχρι να γίνει μια τροποποίηση σε αυτές τις σελίδες.

```
--------------------------------------------------------------------------------
pa=get_physical_address((uint64_t)heap_private_buf);
printf("Physical Address of private heap buffer requested by child: 0x%lx\n",pa);
printf("Vitual Address of private heap buffer in child: ");
show_va_info((uint64_t)heap_private_buf);

pa=get_physical_address((uint64_t)heap_private_buf);
printf("Physical Address of private heap buffer requested by parent is:0x%lx\n",pa);
printf("Virtual Address of private heap buffer in parent: ");
show_va_info((uint64_t)heap_private_buf);
--------------------------------------------------------------------------------
```

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

Physical Address of private heap buffer requested by parent is:0x14ef25000
Virtual Address of private heap buffer in parent: 7f2cb0ecd000-7f2cb0ece000 rw-p 00000000 00:00 0
Physical Address of private heap buffer requested by child: 0x14ef25000
Vitual Address of private heap buffer in child: 7f2cb0ecd000-7f2cb0ece000 rw-p 00000000 00:00 0
```

Ερώτηση 9:

Γράφουμε στον private buffer από τη διεργασία παιδί και τυπώνουμε για άλλη μια φορά τη φυσική και εικονική του διεύθυνση. Παρατηρούμε ότι μετά την επεξεργασία η φυσική διεύθυνση του private buffer της διεργασίας παιδί έχει αλλάξει. Αυτό συμβαίνει διότι, όπως αναφέραμε και προηγουμένως, όταν καλείται η fork η parent και η child διεργασία αρχικά μοιράζονται memory pages με σκοπό την καλύτερη αξιοποίηση των πόρων του συστήματος και μέχρι κάποια από τις διεργασίες προσπαθήσει να γράψει σε κάποια private σελίδα αυτό δεν αλλάζει. Όταν επιχειρήσει μια διεργασία να τροποποιήσει τον private buffer της τότε το λειτουργικό σύστημα κάνει την αντιγραφή σε μια νέα φυσική διεύθυνση και ενημερώνεται ο πίνακας σελίδων. Αυτή η τεχνική ονομάζεται Copy on Write.

```
--------------------------------------------------------------------------------
memset(heap_private_buf,1,buffer_size);//initialize buffer with 1s
pa=get_physical_address((uint64_t)heap_private_buf);
printf("Physical Memory of private heap buffer requested by child: 0x%lx\n",pa);
printf("Virtual Address of private heap buffer in child: ");
show_va_info((uint64_t)heap_private_buf);

pa=get_physical_address((uint64_t)heap_private_buf);
printf("Physical Address of private heap buffer requested by parent is:0x%lx\n",pa);
printf("Virtual Address of private heap buffer in parent: ");
show_va_info((uint64_t)heap_private_buf);
--------------------------------------------------------------------------------
```

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

Physical Address of private heap buffer requested by parent is:0x14ef25000
Virtual Address of private heap buffer in parent: 7f2cb0ecd000-7f2cb0ece000 rw-p 00000000 00:00 0
Physical Memory of private heap buffer requested by child: 0x23e902000
Virtual Address of private heap buffer in child: 7f2cb0ecd000-7f2cb0ece000 rw-p 00000000 00:00 0
```

Ερώτηση 10:

Γράφουμε στον shared buffer από τη διεργασία παιδί και τυπώνουμε τη φυσική και εικονική του διεύθυνση για τις δύο διεργασίες. Παρατηρούμε ότι και η εικονική και η φυσική μνήμη είναι κοινές. Αυτό οφείλεται στο γεγονός ότι ο buffer έχει αρχικοποιηθεί ως shared και επομένως οι δύο διεργασίες μπορούν να "μοιράζονται" τη μνήμη που καταλαμβάνει απεικονίζοντας τον σε κοινή φυσική διεύθυνση.

```
--------------------------------------------------------------------------------
memset(heap_shared_buf,1,buffer_size);
pa=get_physical_address((uint64_t)heap_shared_buf);
printf("Physical Address of shared heap buffer requested by child: 0x%lx\n",pa);
printf("Virtual Address of shared heap buffer in child: ");
show_va_info((uint64_t)heap_private_buf);

pa=get_physical_address((uint64_t)heap_shared_buf);
printf("Physical Address of shared heap buffer requested by child: 0x%lx\n",pa);
printf("Virtual Address of shared heap buffer in child: ");
show_va_info((uint64_t)heap_shared_buf);
--------------------------------------------------------------------------------
```

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

Physical Address of shared heap buffer requested by child: 0x20bc09000
Virtual Address of shared heap buffer in child: 7f2cb0ea2000-7f2cb0ea3000 rw-s 00000000 00:01 12232
  /dev/zero (deleted)
Physical Address of shared heap buffer requested by child: 0x20bc09000
Virtual Address of shared heap buffer in child: 7f2cb0ecd000-7f2cb0ece000 rw-p 00000000 00:00 0
```

Ερώτηση 11:

Αφαιρούμε το δικαίωμα εγγραφής στον shared buffer από το παιδί. Τυπώνουμε τους χάρτες εικονικής μνήμης για τις δύο διεργασίες και παρατηρούμε ότι πλέον ο shared buffer είναι read only στο παιδί.

```
--------------------------------------------------------------------------------
if(mprotect(heap_shared_buf,buffer_size,PROT_READ)==-1){
       perror("mprotect");
       exit(EXIT_FAILURE);}
printf("Virtual Memory map of child after mprotect:\n");
show_maps();
show_va_info((uint64_t)heap_shared_buf);


printf("Virtual Memory map of parent after mprotect:\n");
show_maps();
show_va_info((uint64_t)heap_private_buf);
--------------------------------------------------------------------------------
```

```
Virtual Memory Map of process [1573274]:
55a7dfecf000-55a7dfed0000 r--p 00000000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed0000-55a7dfed1000 r-xp 00001000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed1000-55a7dfed2000 r--p 00002000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed2000-55a7dfed3000 r--p 00002000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed3000-55a7dfed4000 rw-p 00003000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7e18c2000-55a7e18e3000 rw-p 00000000 00:00 0                          [heap]
7f2cb0cc9000-7f2cb0ceb000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0ceb000-7f2cb0e44000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e44000-7f2cb0e93000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e93000-7f2cb0e97000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e97000-7f2cb0e99000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e99000-7f2cb0e9f000 rw-p 00000000 00:00 0
7f2cb0ea2000-7f2cb0ea3000 rw-s 00000000 00:01 12232                      /dev/zero (deleted)
7f2cb0ea3000-7f2cb0ea4000 r--s 00000000 00:26 4330062                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/file.txt
7f2cb0ea4000-7f2cb0ea5000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ea5000-7f2cb0ec5000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ec5000-7f2cb0ecd000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ecd000-7f2cb0ece000 rw-p 00000000 00:00 0
7f2cb0ece000-7f2cb0ecf000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ecf000-7f2cb0ed0000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ed0000-7f2cb0ed1000 rw-p 00000000 00:00 0
7ffdce700000-7ffdce721000 rw-p 00000000 00:00 0                          [stack]
7ffdce7df000-7ffdce7e3000 r--p 00000000 00:00 0                          [vvar]
7ffdce7e3000-7ffdce7e5000 r-xp 00000000 00:00 0                          [vdso]
```

```
Virtual Memory map of child after mprotect:

Virtual Memory Map of process [1573275]:
55a7dfecf000-55a7dfed0000 r--p 00000000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed0000-55a7dfed1000 r-xp 00001000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed1000-55a7dfed2000 r--p 00002000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed2000-55a7dfed3000 r--p 00002000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7dfed3000-55a7dfed4000 rw-p 00003000 00:26 4329813                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/mmap
55a7e18c2000-55a7e18e3000 rw-p 00000000 00:00 0                          [heap]
7f2cb0cc9000-7f2cb0ceb000 r--p 00000000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0ceb000-7f2cb0e44000 r-xp 00022000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e44000-7f2cb0e93000 r--p 0017b000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e93000-7f2cb0e97000 r--p 001c9000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e97000-7f2cb0e99000 rw-p 001cd000 fe:01 144567                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f2cb0e99000-7f2cb0e9f000 rw-p 00000000 00:00 0
7f2cb0ea2000-7f2cb0ea3000 r--s 00000000 00:01 12232                      /dev/zero (deleted)
7f2cb0ea3000-7f2cb0ea4000 r--s 00000000 00:26 4330062                    /home/oslab/oslab019/3rd_lab_report/1st_exerci
se/file.txt
7f2cb0ea4000-7f2cb0ea5000 r--p 00000000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ea5000-7f2cb0ec5000 r-xp 00001000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ec5000-7f2cb0ecd000 r--p 00021000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ecd000-7f2cb0ece000 rw-p 00000000 00:00 0
7f2cb0ece000-7f2cb0ecf000 r--p 00029000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ecf000-7f2cb0ed0000 rw-p 0002a000 fe:01 144563                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f2cb0ed0000-7f2cb0ed1000 rw-p 00000000 00:00 0
7ffdce700000-7ffdce721000 rw-p 00000000 00:00 0                          [stack]
7ffdce7df000-7ffdce7e3000 r--p 00000000 00:00 0                          [vvar]
7ffdce7e3000-7ffdce7e5000 r-xp 00000000 00:00 0                          [vdso]
```

Η αφαίρεση του δικαιώματος αυτού έγινε με χρήση της mprotect σύμφωνα με την οποία αλλάζουν τα δικαιώματα της διεργασίας που την καλεί σύμφωνα με το τρίτο της όρισμα. Πιο αναλυτικά, η διεργασία που καλεί την mprotect έχει πλέον τα δικαιώματα που προσδιορίζονται από το τρίτο της όρισμα για ένα χώρο μνήμης που ξεκινάει από το πρώτο της όρισμα και είναι μεγέθους που καθορίζεται από το δεύτερο.

Ερώτημα 12:

Κάνοντας χρήση της munmap() αποδεσμεύουμε όλους τους buffers στις δύο διεργασίες.

```
--------------------------------------------------------------------------------
if(munmap(heap_private_buf,buffer_size)==-1){
        perror("munmap");
        exit(EXIT_FAILURE);}
if(munmap(heap_shared_buf,buffer_size)==-1){
        perror("munmap");
        exit(EXIT_FAILURE);}
if(munmap(file_shared_buf,buffer_size)==-1){
        perror("munmap");
        exit(EXIT_FAILURE);}


if(munmap(heap_private_buf,buffer_size)==-1){
        perror("munmap");
        exit(EXIT_FAILURE);}
if(munmap(heap_shared_buf,buffer_size)==-1){
        perror("munmap");
        exit(EXIT_FAILURE);}
if(munmap(file_shared_buf,buffer_size)==-1){
        perror("munmap");
        exit(EXIT_FAILURE);}
--------------------------------------------------------------------------------
```

1.2. Παράλληλος υπολογισμός Mandelbrot με διεργασίες αντί για νήματα

1.2.1 Semaphores πάνω από διαμοιραζόμενη μνήμη

Ο κώδικας που χρησιμοποιήθηκε είναι παρόμοιος με τον κώδικα της Άσκησης 2 επομένως θα εξηγηθούν μόνο τα τμήματά του που αφορούν την υλοποίηση με processes αντί για threads.

Το πρόβλημα που δημιουργείται στην υλοποίηση με διεργασίες αντί για νήματα είναι η επικοινωνία μεταξύ τους καθώς δεν μοιράζονται χώρο μνήμης. Προκύπτει επομένως η ανάγκη δημιουργίας μια κοινής μνήμης μεταξύ των διεργασιών έτσι ώστε να μπορούν όλες να έχουν πρόσβαση στα semaphores για να γνωρίζουν εάν πρέπει να περιμένουν ή να εκτελέσουν το κρίσιμο τμήμα τους. Για τον σκοπό αυτό συμπληρώνουμε τη συνάρτηση create_shared_memory_area() με τη mmap και δεσμεύουμε ένα κοινό (MAP_SHARED) ,μη αντιστοιχισμένο με φάκελο (MAP_ANONYMOUS) και προσβάσιμο για ανάγνωση και εγγραφή (PROT_READ|PROT_WRITE) από όλες τις διεργασίες τμήμα μνήμης. Επιπλέον, χρησιμοποιείται και η συνάρτηση destroy_shared_memory_area() για να γίνει ορθά η αποδέσμευση της μνήμης. Όσο για την χρήση των semaphores, εφόσον ισχύει και για τις διεργασίες ο ίδιος τρόπος διαμοιρασμού των υπολογισμών ο κώδικας για τη χρήση είναι πανομοιότυπος με αυτόν της Άσκησης 2.

Ο πηγαίος κώδικας βρίσκεται στο αρχείο ***mandel-fork-sem.c***

Ερώτηση 1

Περιμένουμε η υλοποίηση με threads να είναι πιο αποδοτική από την υλοποίηση με διεργασίες για τους εξής λόγους:

Απομόνωση Μνήμης και Κοινή μνήμη: Γνωρίζουμε ότι κάθε νήμα χρησιμοποιεί τον ίδιο χώρο διευθύνσεων, οπότε όλα επενεργούν στα ίδια δυναμικά και στατικά δεδομένα. Οι διεργασίες παρόλο που κανονικά διαθέτουν ιδιωτικό χώρο μνήμης μπορούν να επιτύχουν το ίδιο αποτέλεσμα με τη χρήση ενός κοινού τμήματος μνήμης. Η απεικόνιση όμως αυτού του τμήματος στον χώρο διευθύνσεων της κάθε διεργασίας προσθέτει χρονικές επιβαρύνσεις που δεν υπάρχουν στην υλοποίηση με threads.

Context Switching: Η εναλλαγή μεταξύ διεργασιών είναι εμφανώς δυσκολότερη και πιο χρονοβόρα απο την εναλλαγή μεταξύ νημάτων καθώς εκτός από την αποθήκευση του PC και των καταχωρητών θα πρέπει για να γίνει το context switch να αντικατασταθεί και το process control block που αποθηκεύει όλες τις πληροφορίες της διεργασίας.

Η ύπαρξη των semaphores στη διαμοιραζόμενη μνήμη μεταξύ διεργασιών είναι απαραίτητη για το σωστό συγχρονισμό τους όμως προκαλεί καθυστερήσεις στην εκτέλεση των waiting διεργασιών. Είναι προφανές πως για να παραχθεί το επιθυμητό αποτέλεσμα πρέπει να εκτελεστούν οι διεργασίες με τη σειρά ανάλογα με τις γραμμές που υπολογίζουν γεγονός που αυξάνει τον χρόνο εκτέλεσης του προγράμματος αλλά διασφαλίζει την ακρίβεια που ζητείται.

1.2.2 Υλοποίηση χωρίς Semaphores

Ερώτηση 1

Στην υλοποίηση με χρήση buffer δεν χρησιμοποιείται κάποιος μηχανισμός συγχρονισμού κατά τον υπολογισμό των γραμμών καθώς το κρίσιμο τμήμα είναι η τελική απεικόνισή τους την οποία διαχειρίζεται η αρχική διεργασία. Επομένως, τον συγχρονισμό αναλαμβάνει η parent διεργασία η οποία περιμένει να τερματίσουν όλες οι διεργασίες παιδιά και ύστερα αφού έχουν αποθηκευτεί οι κατάλληλες τιμές στις γραμμές του buffer τις διατρέχει και τις εκτυπώνει σταδιακά. Η υλοποίηση αυτή δίνει σωστό αποτέλεσμα διότι ο buffer έχει χρησιμοποιηθεί με τέτοιο τρόπο (συνάρτηση buf_compute_mandel_line) έτσι ώστε κάθε του i γραμμή να αντιστοιχεί στην i γραμμή του σχήματος επομένως τα παιδιά θα υπολογίσουν ασύγχρονα τις τιμές που θα αποθηκευτούν στις γραμμές του buffer και ο γονέας θα τις εκτυπώσει με τη σειρά για την επίτευξη της τελικής ορθής απεικόνισης. Τέλος, όπως και πριν θα χρησιμοποιηθεί η destroy_shared_memory_area() για να αποδεσμευτεί το δισδιάστατο τμήμα μνήμης που χρησιμοποιήθηκε.

Αν είχαμε μικρότερο buffer (NPROCS x x_chars) δεν θα ήταν δυνατή η χρήση της ίδια λογικής για την υλοποίηση του προβλήματος. Θα έπρεπε η απεικόνιση του αποτελέσματος να γινόταν σταδιακά ανά NPROCS γραμμές. Αυτό θα μπορούσε να επιτευχθεί αν η κάθε διεργασία μετά τον υπολογισμό ΜΙΑΣ γραμμής κάνει raise(SIGSTOP) και στέλνει σήμα στην calling η οποία με τη σειρά της θα αναμένει NPROCS σήματα και αφού τα λάβει θα κάνει το output του buffer και ύστερα θα ξυπνάει συνεχίζει μια μια τις διεργασίες.

Ο πηγαίος κώδικας βρίσκεται στο αρχείο ***mandel-fork.c***