Project 2:
Memory Management Tools

Selma Karasoftić (23004204)
Amina Brković (23004548)

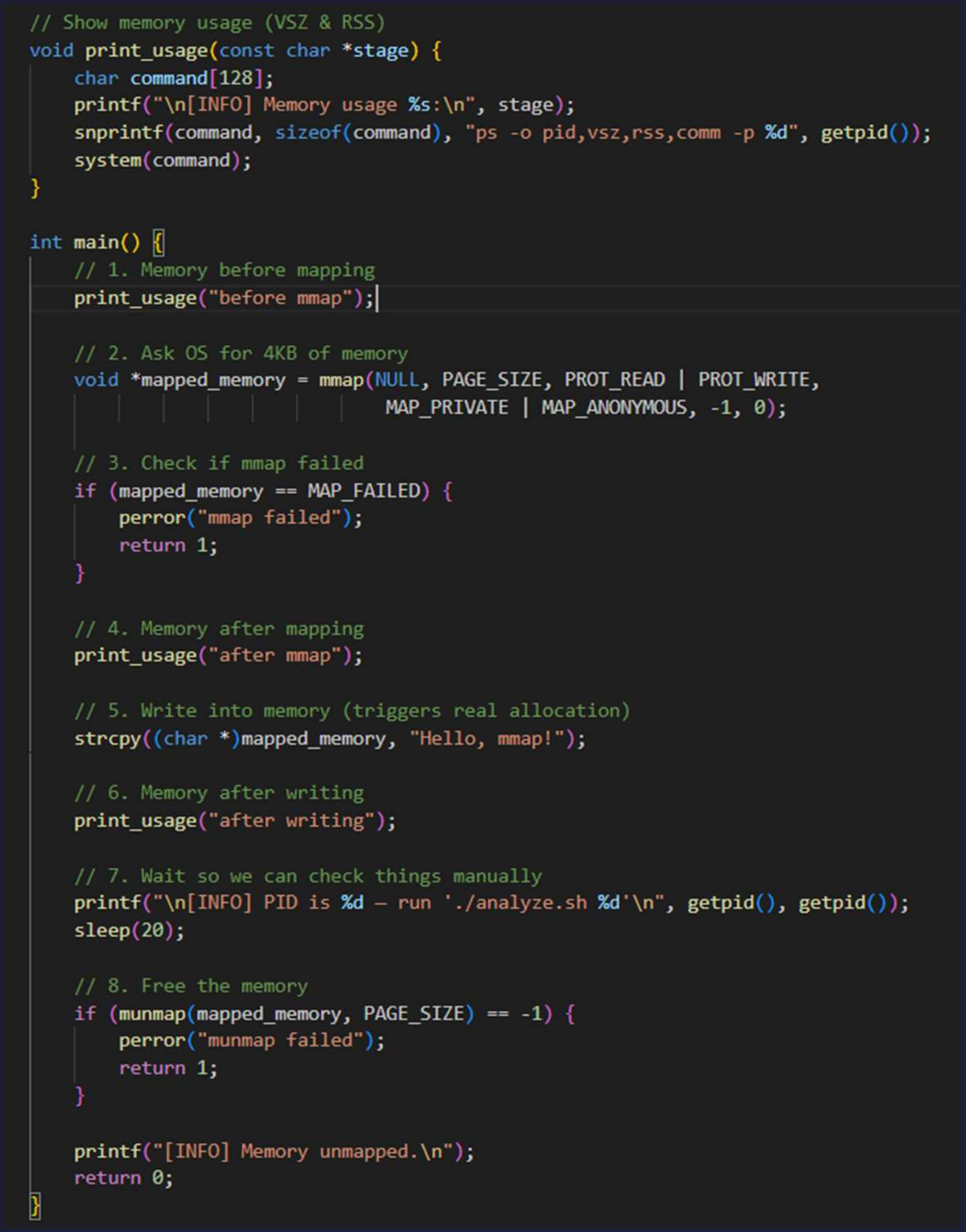
IT 204 – Operating Systems

Summary of Tasks

- Learn how *mmap* and *munap* system calls work
- Implement a C program for *mmap* and *munap*
- Track VSZ and RSS using ps at three stages: before mmap, after mmap, and after writing.
- Learn about lazy allocation, analyze /proc/<PID>/maps, interpret memory behavior.
- Create a shell script (analyze.sh) to automate memory inspection.

Code Explanation – mmap_demo.c

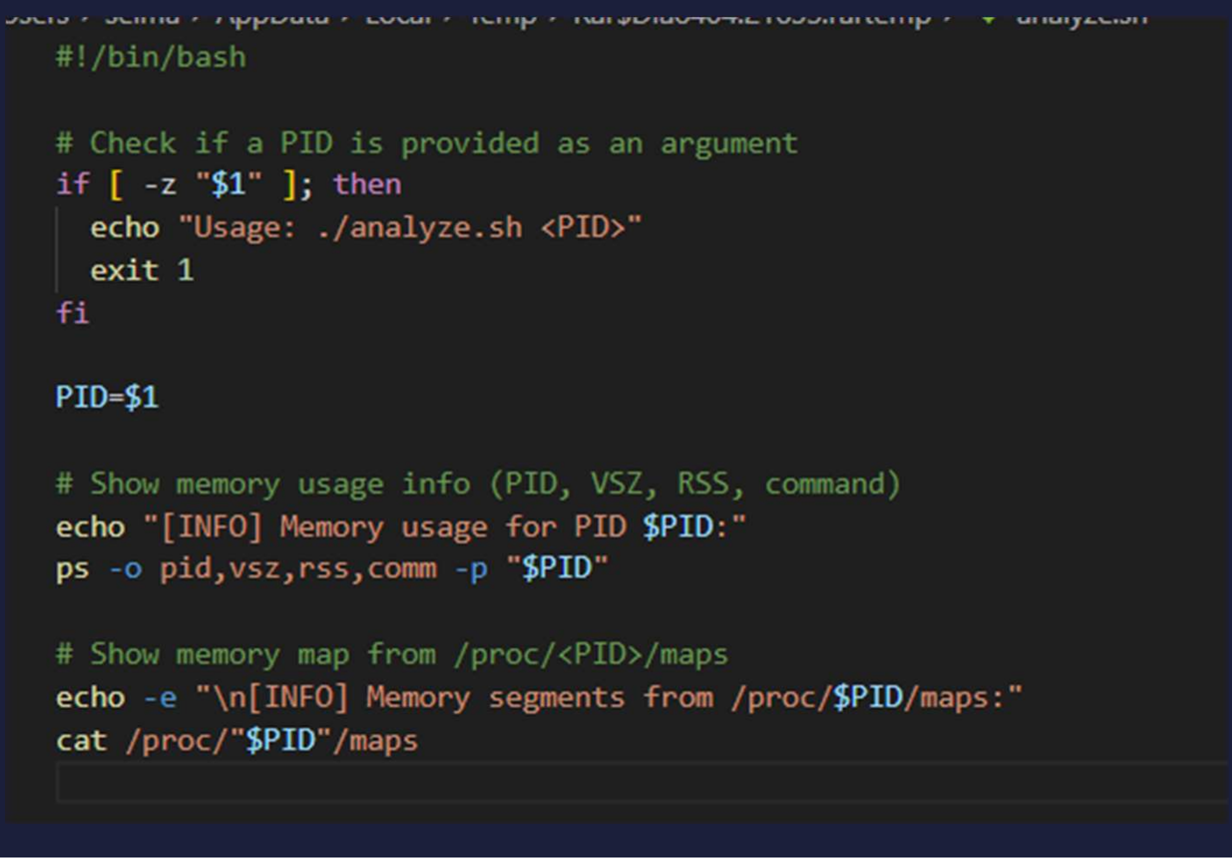
- Used mmap() to allocate 4KB anonymous memory.
- Tracked memory before, after mmap, and after writing.
- Wrote 'Hello, mmap!' to trigger allocation.
- Slept 20 seconds for manual inspection.
- mapped memory using munmap().



```c
// Show memory usage (VSZ & RSS)
void print_usage(const char *stage) {
    char command[128];
    printf("\n[INFO] Memory usage %s:\n", stage);
    snprintf(command, sizeof(command), "ps -o pid,vsz,rss,comm -p %d", getpid());
    system(command);
}

int main() {
    // 1. Memory before mapping
    print_usage("before mmap");

    // 2. Ask OS for 4KB of memory
    void *mapped_memory = mmap(NULL, PAGE_SIZE, PROT_READ | PROT_WRITE,
                                MAP_PRIVATE | MAP_ANONYMOUS, -1, 0);

    // 3. Check if mmap failed
    if (mapped_memory == MAP_FAILED) {
        perror("mmap failed");
        return 1;
    }

    // 4. Memory after mapping
    print_usage("after mmap");

    // 5. Write into memory (triggers real allocation)
    strcpy((char *)mapped_memory, "Hello, mmap!");

    // 6. Memory after writing
    print_usage("after writing");

    // 7. Wait so we can check things manually
    printf("\n[INFO] PID is %d – run './analyze.sh %d'\n", getpid(), getpid());
    sleep(20);

    // 8. Free the memory
    if (munmap(mapped_memory, PAGE_SIZE) == -1) {
        perror("munmap failed");
        return 1;
    }

    printf("[INFO] Memory unmapped.\n");
    return 0;
}
```

Shell Script – analyze.sh

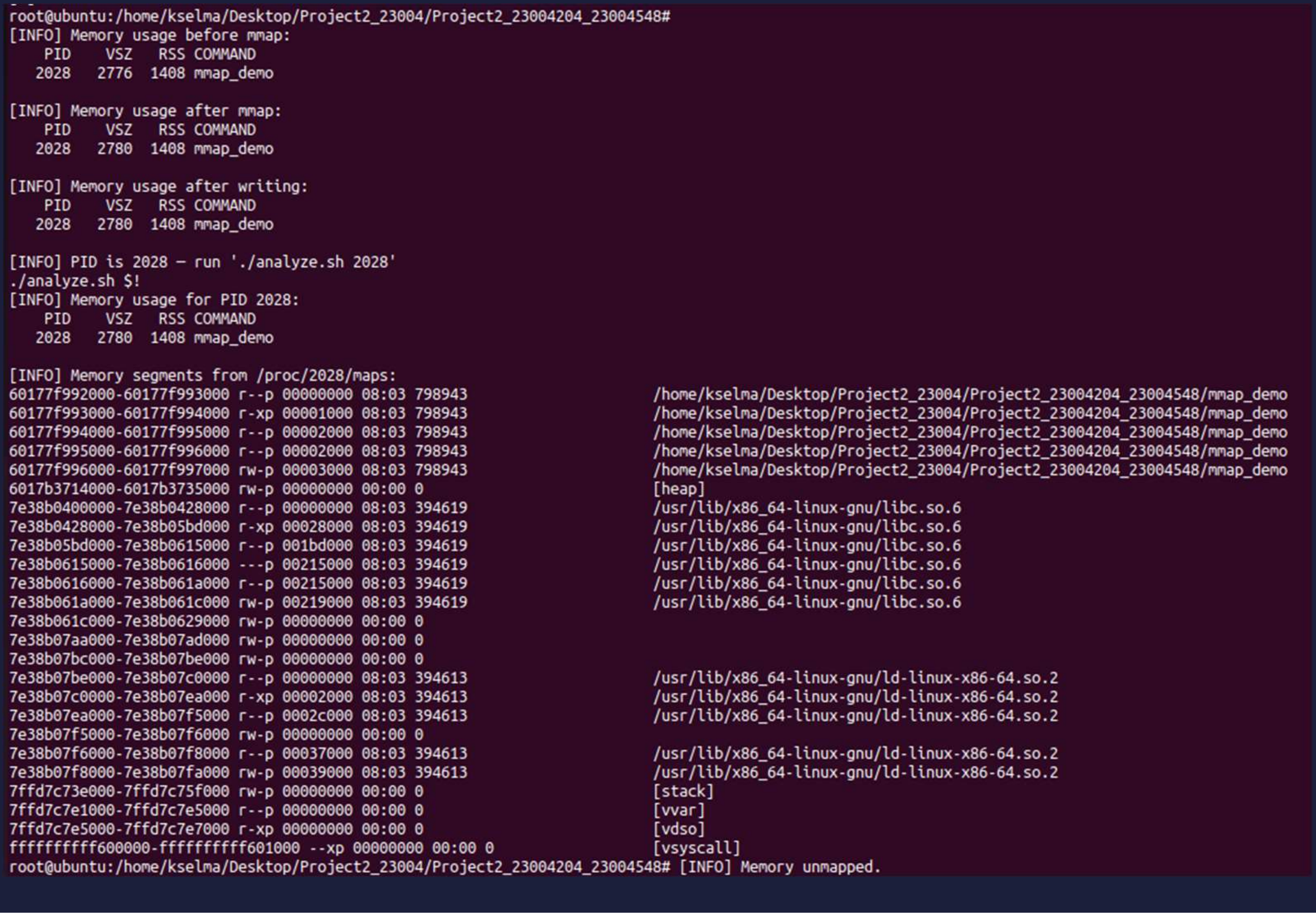
- Accepts a PID as argument.
- Uses ps to show PID, VSZ, RSS.
- Reads memory segments from /proc/<PID>/maps.
- Helps confirm mmap region.
- Includes usage message and comments.


Users › selma › AppData › Local › Temp › Rar$DIa0404.21055.rartemp › analyze.sh

```
#!/bin/bash

# Check if a PID is provided as an argument
if [ -z "$1" ]; then
  echo "Usage: ./analyze.sh <PID>"
  exit 1
fi

PID=$1

# Show memory usage info (PID, VSZ, RSS, command)
echo "[INFO] Memory usage for PID $PID:"
ps -o pid,vsz,rss,comm -p "$PID"

# Show memory map from /proc/<PID>/maps
echo -e "\n[INFO] Memory segments from /proc/$PID/maps:"
cat /proc/"$PID"/maps
```

Output and Observations

- • VSZ increased after mmap (2776 KB → 2780 KB).
- • RSS remained unchanged (1408 KB).
- • mmap region found in /proc/<PID>/maps.



```
root@ubuntu:/home/kselma/Desktop/Project2_23004/Project2_23004204_23004548#
[INFO] Memory usage before mmap:
    PID    VSZ   RSS COMMAND
   2028   2776  1408 mmap_demo

[INFO] Memory usage after mmap:
    PID    VSZ   RSS COMMAND
   2028   2780  1408 mmap_demo

[INFO] Memory usage after writing:
    PID    VSZ   RSS COMMAND
   2028   2780  1408 mmap_demo

[INFO] PID is 2028 — run './analyze.sh 2028'
./analyze.sh $!
[INFO] Memory usage for PID 2028:
    PID    VSZ   RSS COMMAND
   2028   2780  1408 mmap_demo

[INFO] Memory segments from /proc/2028/maps:
60177f992000-60177f993000 r--p 00000000 08:03 798943                     /home/kselma/Desktop/Project2_23004/Project2_23004204_23004548/mmap_demo
60177f993000-60177f994000 r-xp 00001000 08:03 798943                     /home/kselma/Desktop/Project2_23004/Project2_23004204_23004548/mmap_demo
60177f994000-60177f995000 r--p 00002000 08:03 798943                     /home/kselma/Desktop/Project2_23004/Project2_23004204_23004548/mmap_demo
60177f995000-60177f996000 r--p 00002000 08:03 798943                     /home/kselma/Desktop/Project2_23004/Project2_23004204_23004548/mmap_demo
60177f996000-60177f997000 rw-p 00003000 08:03 798943                     /home/kselma/Desktop/Project2_23004/Project2_23004204_23004548/mmap_demo
6017b3714000-6017b3735000 rw-p 00000000 00:00 0                          [heap]
7e38b0400000-7e38b0428000 r--p 00000000 08:03 394619                     /usr/lib/x86_64-linux-gnu/libc.so.6
7e38b0428000-7e38b05bd000 r-xp 00028000 08:03 394619                     /usr/lib/x86_64-linux-gnu/libc.so.6
7e38b05bd000-7e38b0615000 r--p 001bd000 08:03 394619                     /usr/lib/x86_64-linux-gnu/libc.so.6
7e38b0615000-7e38b0616000 ---p 00215000 08:03 394619                     /usr/lib/x86_64-linux-gnu/libc.so.6
7e38b0616000-7e38b061a000 r--p 00215000 08:03 394619                     /usr/lib/x86_64-linux-gnu/libc.so.6
7e38b061a000-7e38b061c000 rw-p 00219000 08:03 394619                     /usr/lib/x86_64-linux-gnu/libc.so.6
7e38b061c000-7e38b0629000 rw-p 00000000 00:00 0
7e38b07aa000-7e38b07ad000 rw-p 00000000 00:00 0
7e38b07bc000-7e38b07be000 rw-p 00000000 00:00 0
7e38b07be000-7e38b07c0000 r--p 00000000 08:03 394613                     /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7e38b07c0000-7e38b07ea000 r-xp 00002000 08:03 394613                     /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7e38b07ea000-7e38b07f5000 r--p 0002c000 08:03 394613                     /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7e38b07f5000-7e38b07f6000 rw-p 00000000 00:00 0
7e38b07f6000-7e38b07f8000 r--p 00037000 08:03 394613                     /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7e38b07f8000-7e38b07fa000 rw-p 00039000 08:03 394613                     /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7ffd7c73e000-7ffd7c75f000 rw-p 00000000 00:00 0                          [stack]
7ffd7c7e1000-7ffd7c7e5000 r--p 00000000 00:00 0                          [vvar]
7ffd7c7e5000-7ffd7c7e7000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                  [vsyscall]
root@ubuntu:/home/kselma/Desktop/Project2_23004/Project2_23004204_23004548# [INFO] Memory unmapped.
```

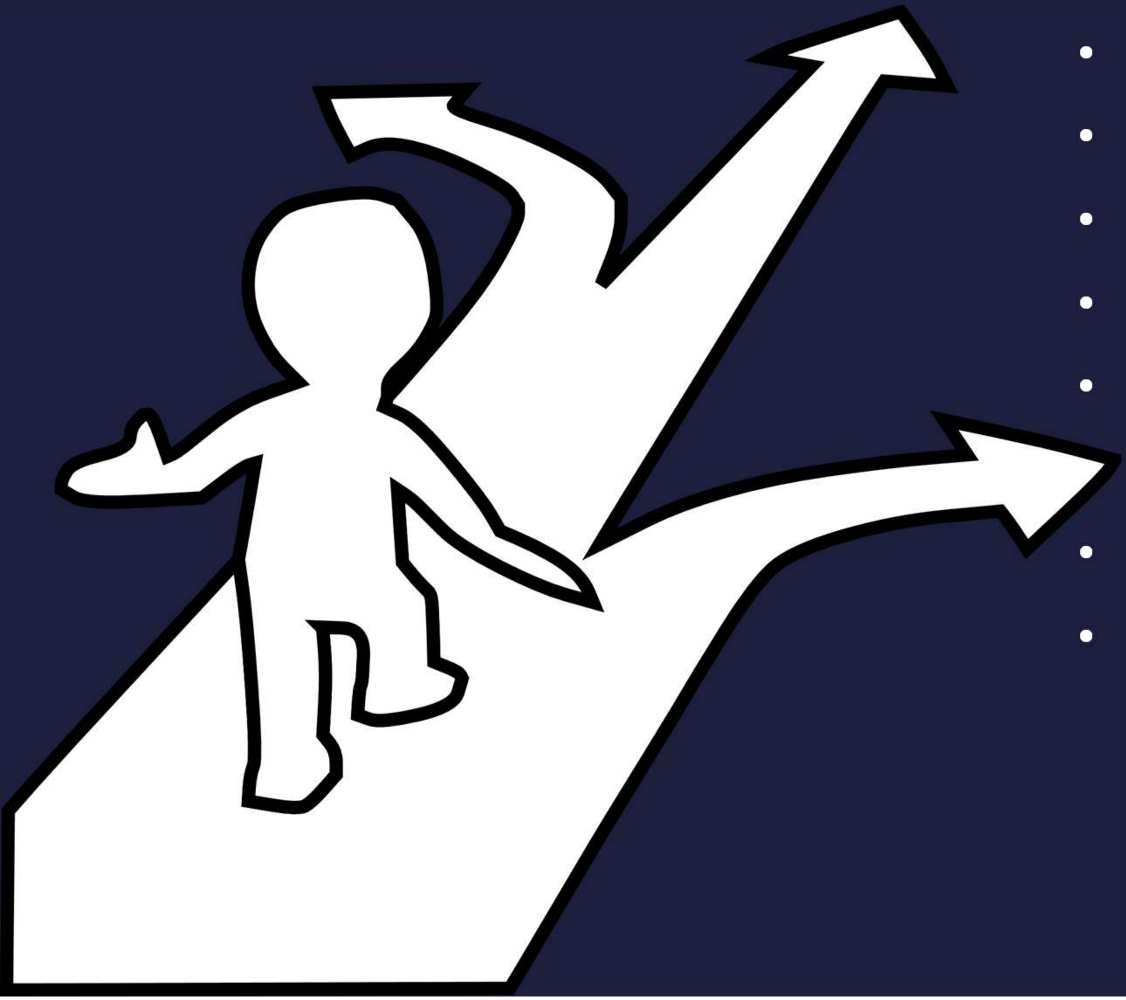
• Anonymous region shown with rw-p and 00:00 0.

• Suggests lazy allocation or system optimization.

How to Compile and Run

- gcc mmap_demo.c -o mmap_demo
- ./mmap_demo &
- ./analyze.sh $!
- chmod +x analyze.sh

Challenges and Our Approach



- RSS didn't increase as expected after writing → confusing at first.
- Debugged with:
- Extra prints
- Multiple test runs
- Changed memory sizes
- Compared script versions
- Realized:

RSS might not update instantly

- Linux can reuse or preallocate memory
- Learned to cross-check with /proc/<PID>/maps instead of trusting only ps.

Conclusion

- Learned about lazy allocation and page faults.
- Analyzed system memory with ps and /proc.
- Understood how mmap works internally.
- Improved scripting and debugging skills.

Thank You for Your Attention!