

# **System call**

**IMPLEMENTATION**

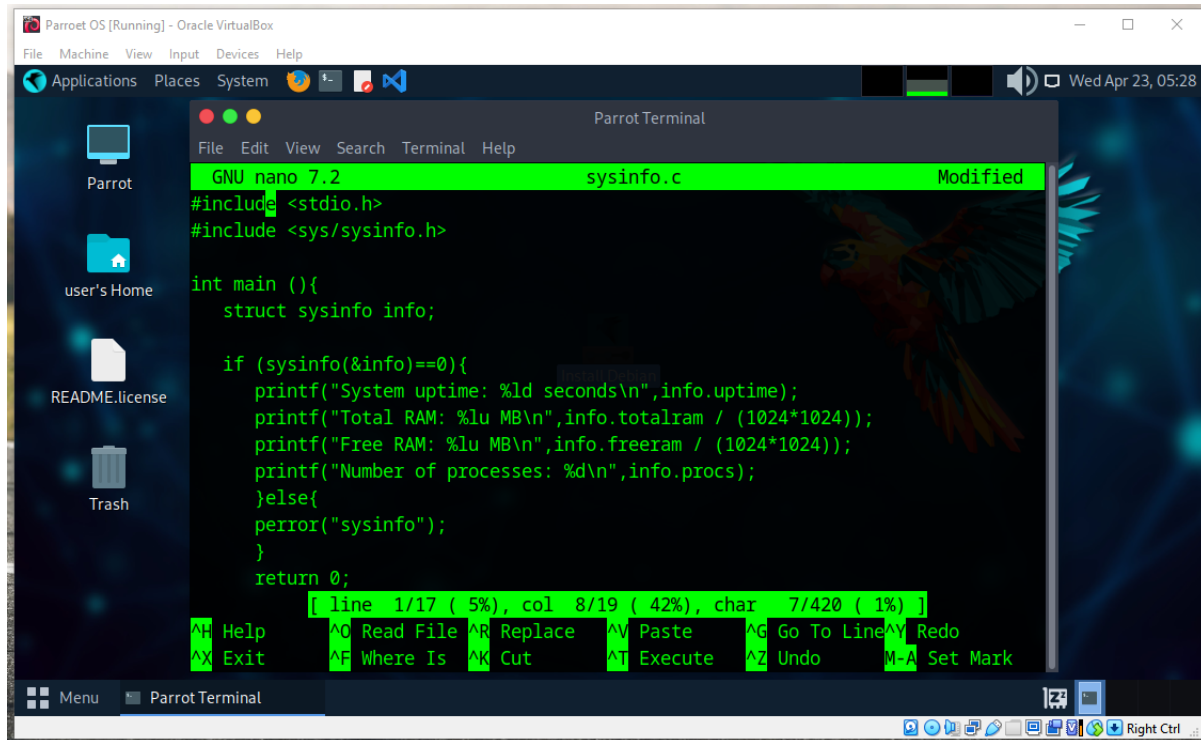
## **System Call Implementation**

In this section, we delve into the implementation of the `sysinfo()` system call by writing and executing a C program within the Parrot OS environment. The `sysinfo()` function, a part of the Linux system call interface, plays a crucial role in accessing various system-level statistics. It is used to retrieve vital information, including the system's uptime (i.e., how long the system has been running since its last boot), the total and available physical RAM, and the number of currently running processes. These metrics provide an overview of the system's resource usage and operational status.

By leveraging `sysinfo()`, a program can directly interface with kernel-maintained data structures, thus allowing user-space applications to query and report on the health and state of the system without needing to access privileged kernel space directly. This kind of interaction exemplifies one of the core principles of system programming—enabling communication between user-level processes and kernel-level resources in a secure and structured manner.

Implementing this system call within a controlled Linux environment such as Parrot OS not only reinforces the conceptual understanding of system-level programming but also provides practical insights into how operating systems manage hardware resources and process control. Furthermore, this exercise highlights the importance of system calls in enabling user applications to perform low-level operations and underscores the relevance of understanding these mechanisms in the context of Operating Systems and System Programming (OSSP) coursework.

By completing this implementation, students gain hands-on experience in interpreting and working with real-time system data, which is essential for anyone pursuing a deeper knowledge of how operating systems function under the hood.

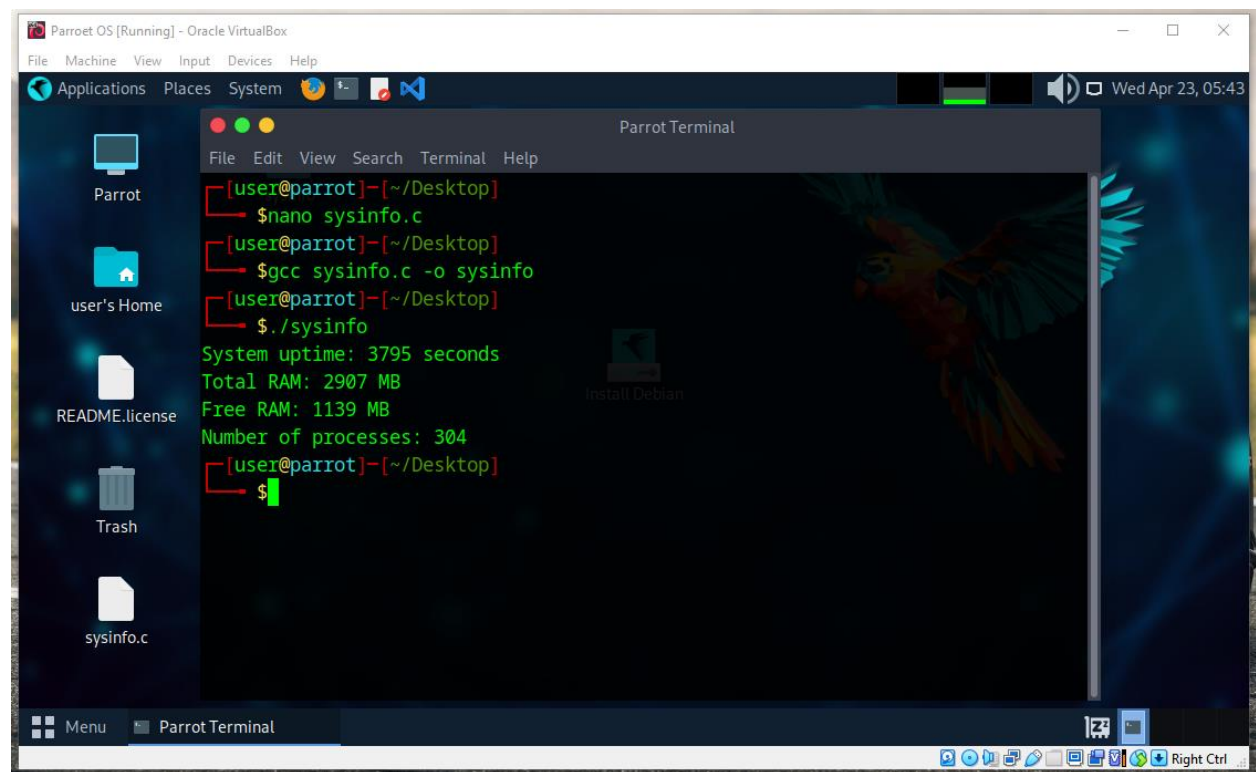


The screenshot shows the Parrot OS desktop environment within an Oracle VM VirtualBox window. The desktop background is a blue abstract pattern with a parrot. On the left sidebar, there are icons for 'Parrot', 'user's Home', 'README.license', and 'Trash'. A 'Parrot Terminal' window is open in the foreground, displaying the GNU nano 7.2 text editor. The editor is editing a file named 'sysinfo.c'. The code in the editor includes headers for `<stdio.h>` and `<sys/sysinfo.h>`, and a `main` function that uses `sysinfo` to print system statistics like uptime, total RAM, free RAM, and the number of processes. The status bar at the bottom of the terminal window shows the current cursor position: line 1/17 (5%), column 8/19 (42%), and character 7/420 (1%).

```
GNU nano 7.2 sysinfo.c Modified
#include <stdio.h>
#include <sys/sysinfo.h>

int main (){
    struct sysinfo info;

    if (sysinfo(&info)==0){
        printf("System uptime: %ld seconds\n",info.uptime);
        printf("Total RAM: %lu MB\n",info.totalram / (1024*1024));
        printf("Free RAM: %lu MB\n",info.freeram / (1024*1024));
        printf("Number of processes: %d\n",info.procs);
    }else{
        perror("sysinfo");
    }
    return 0;
}
[ line 1/17 ( 5%), col 8/19 ( 42%), char 7/420 ( 1%) ]
^H Help      ^O Read File ^R Replace   ^V Paste     ^G Go To Line^Y Redo
^X Exit      ^F Where Is  ^K Cut       ^T Execute   ^Z Undo      M-A Set Mark
```



This screenshot shows the same Parrot OS desktop environment, but the 'Parrot Terminal' window now displays a series of commands and their outputs. The user has navigated to the desktop directory, created the `sysinfo.c` file with `nano`, compiled it with `gcc`, and executed it. The output of the program shows system statistics: 3795 seconds of uptime, 2907 MB of total RAM, 1139 MB of free RAM, and 304 processes. The terminal window also shows the file `sysinfo.c` has been created on the desktop.

```
[user@parrot]~/Desktop
$ nano sysinfo.c
[user@parrot]~/Desktop
$ gcc sysinfo.c -o sysinfo
[user@parrot]~/Desktop
$ ./sysinfo
System uptime: 3795 seconds
Total RAM: 2907 MB
Free RAM: 1139 MB
Number of processes: 304
[user@parrot]~/Desktop
$
```