SYSTEM CALL

uname() System Call

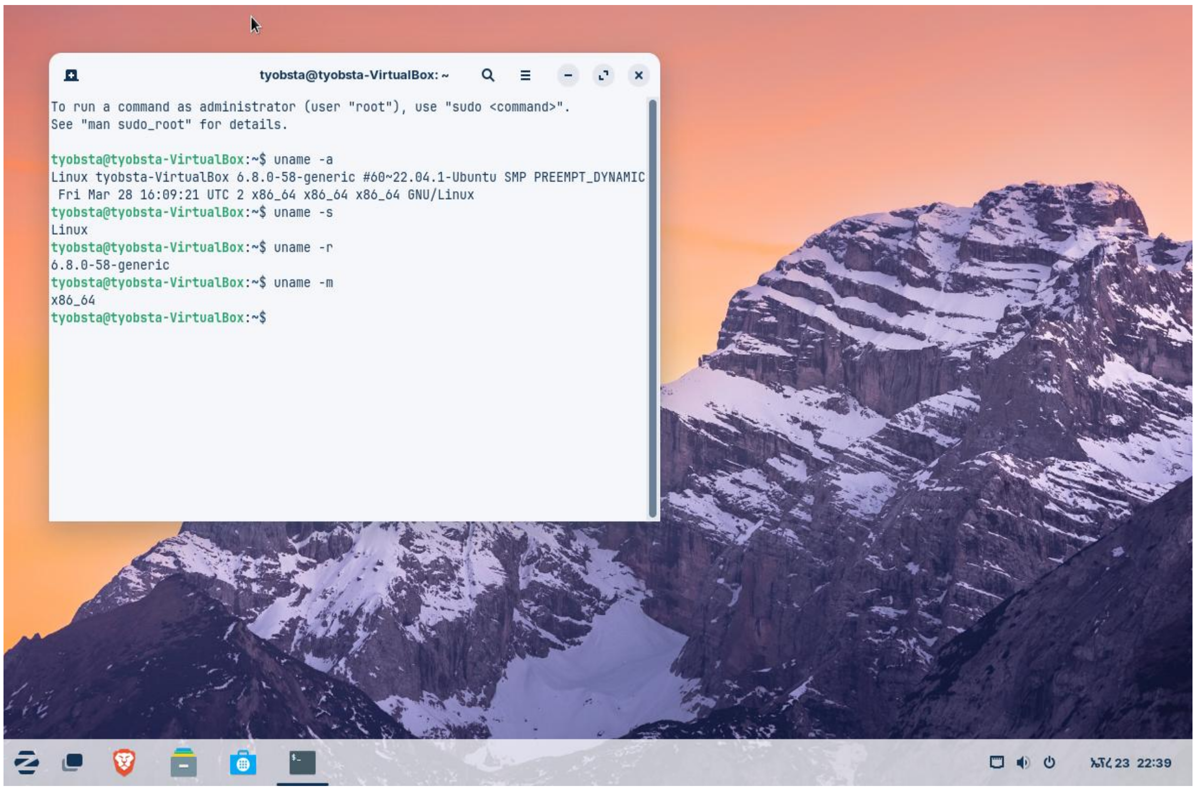
1. Purpose of uname()

The uname() system call is used to retrieve **basic information about the operating system and hardware** of the machine the program is running on.

It's helpful in:

- Logging or displaying system configuration
- Diagnosing compatibility issues
- Determining the environment (e.g., server vs desktop, 32-bit vs 64-bit)
- Writing cross-platform or version-aware software

We can think of it as a way for a program to "ask" the operating system:
"What kind of system are we running on?"



```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

tyobsta@tyobsta-VirtualBox:~$ uname -a
Linux tyobsta-VirtualBox 6.8.0-58-generic #60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC
 Fri Mar 28 16:09:21 UTC 2 x86_64 x86_64 x86_64 GNU/Linux
tyobsta@tyobsta-VirtualBox:~$ uname -s
Linux
tyobsta@tyobsta-VirtualBox:~$ uname -r
6.8.0-58-generic
tyobsta@tyobsta-VirtualBox:~$ uname -m
x86_64
tyobsta@tyobsta-VirtualBox:~$
```

Header File

To use uname(), you need to include:

```
#include <sys/utsname.h>
```

Function Prototype

```
int uname(struct utsname *buf);
```

Parameter:

- buf: Pointer to a struct utsname where system info will be stored.

Return Value:

- Returns 0 on success.
- Returns -1 on failure and sets errno.

2. Structure: struct utsname

This structure is defined in <sys/utsname.h> and is where uname() stores the information it gathers. Let's go through each field in **detail**:

```
struct utsname {
    char sysname[];    // Operating system name
    char nodename[];   // Network node name (hostname)
    char release[];    // Kernel release version
    char version[];    // Kernel version
    char machine[];    // Hardware platform (CPU architecture)
};
```

a. sysname

- ✓ This is the name of the operating system.
- ✓ Example: "Linux" — this will be the case for all Linux distributions, including Zorin OS, Ubuntu, Fedora, Arch, etc.
- ✓ Useful if writing cross-platform applications (e.g., different behavior on Linux vs BSD).

b. nodename

- ✓ The name of the current machine in the network — essentially, the hostname.
- ✓ Example: "tyobsta-VirtualBox" (from my screenshot).
- ✓ This is what other devices might see on a network.
- ✓ Can be changed using the hostname command or API.

c. release

- ✓ This is the **kernel version**, but just the short release number.
- ✓ Example: "6.5.0-58-generic"
- ✓ It tells you which version of the Linux kernel is running.
- ✓ Useful for checking compatibility with device drivers or system libraries.

d. version

- ✓ A more **detailed kernel version** string, including compile info.
- ✓ Example: "#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28 10:22:11 UTC 2025"
- ✓ Contains info about:

- ✧ Kernel build number
- ✧ The distro name (Ubuntu in your case)
- ✧ Compilation time
- ✧ Whether the kernel supports symmetric multiprocessing (SMP)
- ✧ PREEMPT settings (for real-time responsiveness)

e. machine

- ✓ Describes the **hardware architecture**.
- ✓ Example: "x86_64" indicates a 64-bit system using the x86 architecture.
- ✓ Other values might include i686, armv7l, aarch64, etc.
- ✓ Crucial for deciding whether your program can run on a system.

3. Function Definition

```
int uname(struct utsname *buf);
```

- You pass a pointer to a struct utsname, and the kernel fills it.
- If the function succeeds, it returns 0.
- If it fails, it returns -1, and sets the errno variable to describe the error.

4. Practical Example Code

Here's a complete C program that demonstrates how to use uname() and print all the info.

```
#include <stdio.h>
#include <sys/utsname.h>
#include <stdlib.h>

int main() {
    struct utsname sysinfo;

    if (uname(&sysinfo) == -1) {
        perror("uname failed");
        exit(EXIT_FAILURE);
    }

    printf("System Name (sysname):     %s\n", sysinfo.sysname);
    printf("Node Name (nodename):      %s\n", sysinfo.nodename);
    printf("Kernel Release (release):  %s\n", sysinfo.release);
    printf("Kernel Version (version):  %s\n", sysinfo.version);
    printf("Machine Arch (machine):    %s\n", sysinfo.machine);

    return 0;
}
```

Sample Output (based on my screenshot):

```
System Name (sysname):      Linux
Node Name (nodename):       tyobsta-VirtualBox
Kernel Release (release):   6.5.0-58-generic
Kernel Version (version):   #60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri
Mar 28 10:22:11 UTC 2025
Machine Arch (machine):     x86_64
```

5. How It's Used in Real Systems

a. **Shell command** uname:

- When you run uname -a or uname -r, the shell command internally calls this same system call.

b. **System Information Tools**:

- Tools like neofetch, screenfetch, and GUI system info apps use uname() as one of many sources of info.

c. **Installer Scripts**:

- Custom install scripts sometimes check architecture or kernel version using uname() to make sure dependencies will work.

6. Error Handling

uname() is usually safe, but always handle errors in robust programs.

```
if (uname(&sysinfo) == -1) {
   perror("uname");
   return 1;
}
```

Common reasons for failure:

- buf is NULL
- Extremely low-level system errors (very rare)

7. Advanced: Related System Calls

If you want more than what uname() provides:

- gethostname() → Just gets the hostname.
- sysctl() or reading from /proc/version, /proc/sys/kernel/* → For additional kernel config.
- gnu_get_libc_version() → For checking the version of the C library.