**Linux and Windows**

There are three main Linux commands: cp which stands for copy, rm which stands for remove / delete, and cat which stands for concatenate. All three of these commands are system calls of type file management.

CP:

1. This command copies a file or a group of files in different ways.
2. It can copy a file into a new file created through the command.
3. It can copy a file into another directory.

Two main flags:

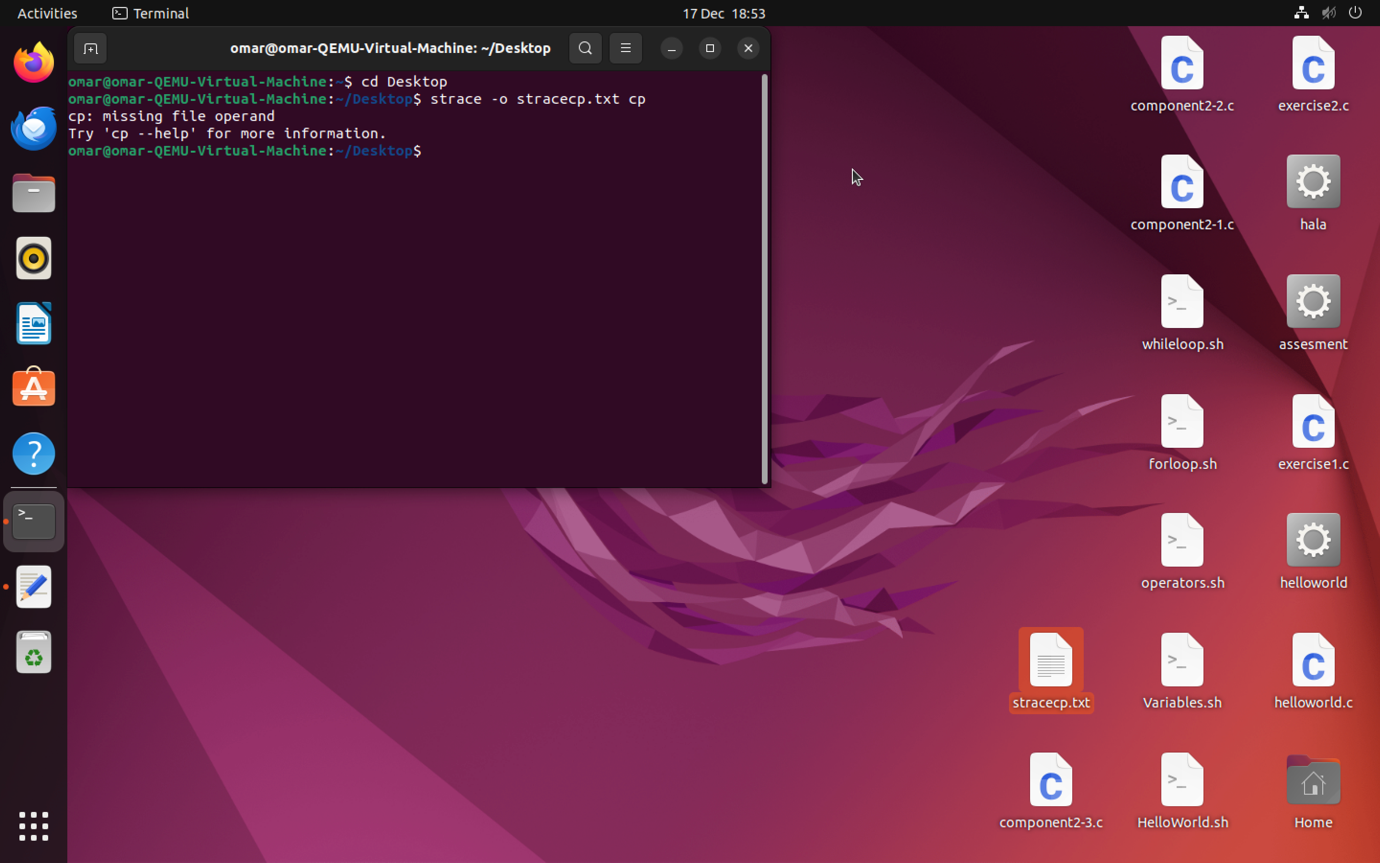
1. -i which warns the user before overwriting the destination file.
2. -R which can copy an entire directory structure into another.

Cp syntax:

1. cp file1.txt file2.txt: copies file1 content in to file2, if file2 does not exist it will be created with the name file2. And if file2 already exists it will be overwritten.
2. cp -i file1.txt file2.txt on the next line you can either respond with “y” for yes / confirm or “n” for no to not carry on with the task
3. cp -R Downloads Desktop copies the whole directory of Downloads into Desktop

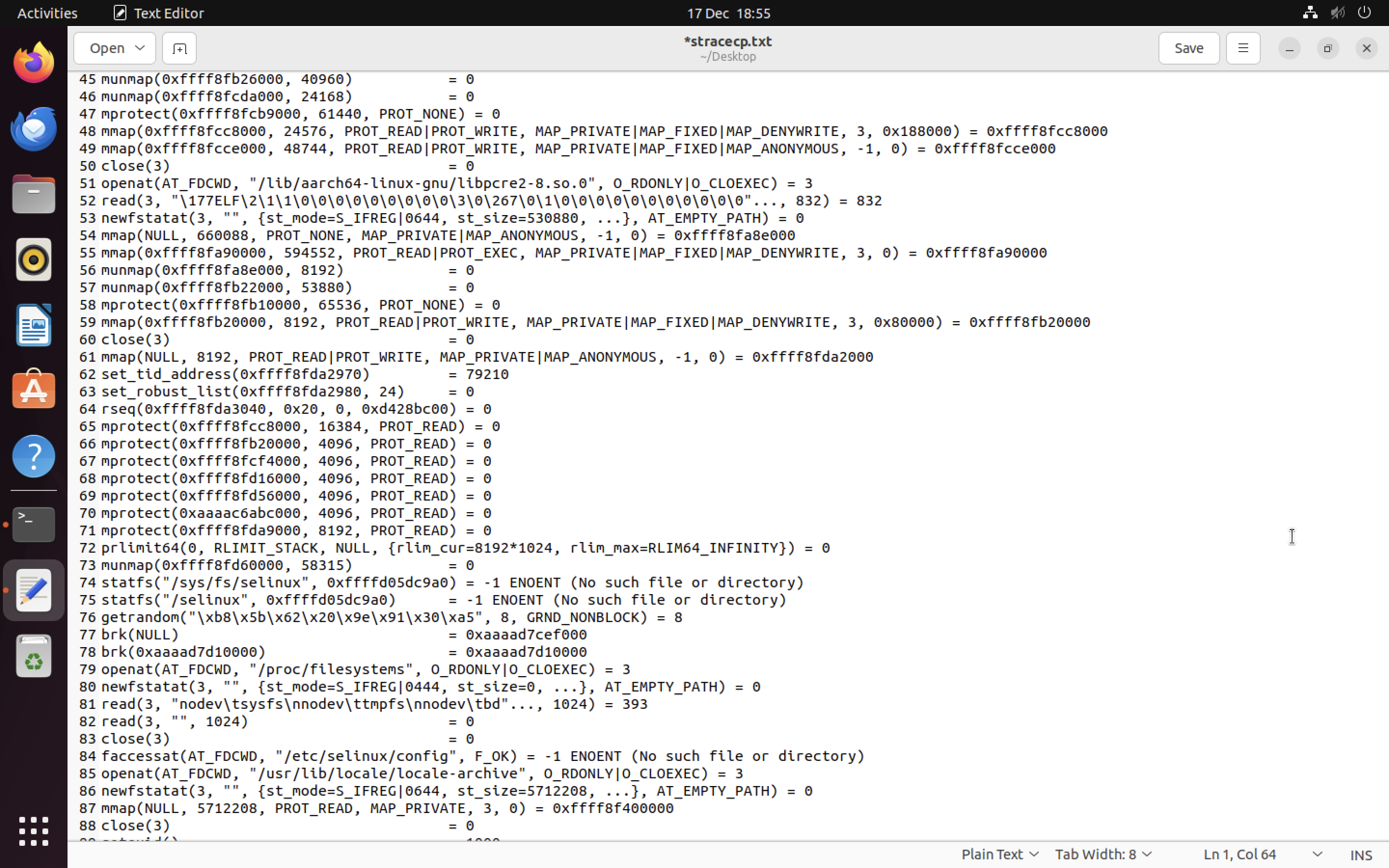
GeeksforGeeks, (2019)

I used strace to see how cp works



I saved the output in a text file to read it easier







Strace shows the main system calls triggered to perform the commands there will be a table after the screenshots of all straces for the three commands, showing what system calls were triggered by which command.

rm:

1. This command deletes or removes one or more files
2. It operates silently i.e., deletes without confirming
3. Once a file is deleted it cannot be recovered
4. Can delete directories if they are empty

Three main flags:

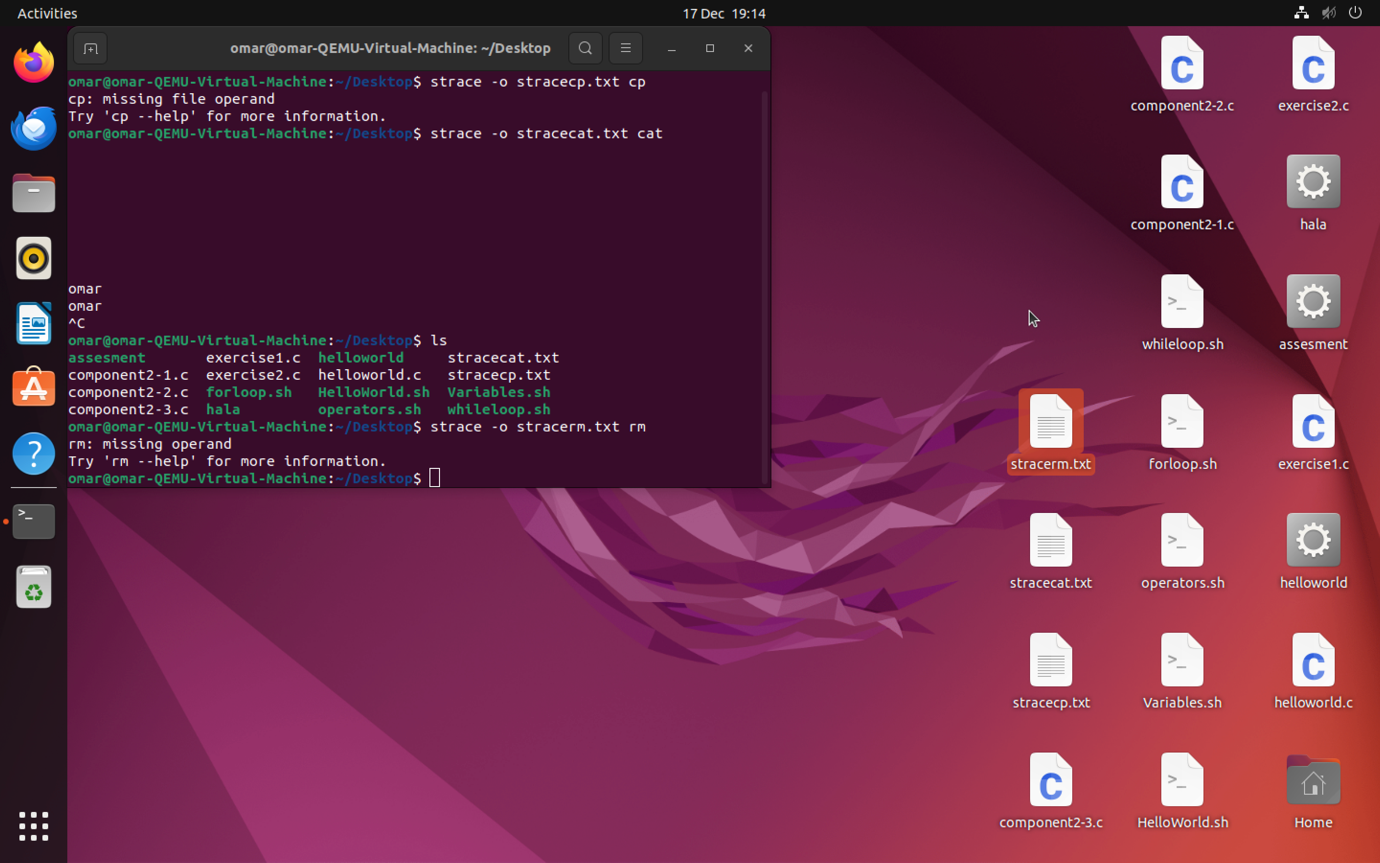
1. -i makes the command ask the user for confirmation before removing each file
2. -R or -r performs a recursive search for all subdirectories and files within these subdirectories. At every stage, it deletes everything it finds. This command also enables the user to delete directories even if they have files in them.
3. -f force removal, removes a file without any prompt even if it is a write protected file

rm syntax:

1. rm file1.txt this deletes file1
2. rm file1.txt file2.txt this deletes file1 and file2
3. rm -i file1.txt on the next line you can either respond with “y” for yes / confirm or “n” for no to not carry on with the task
4. rm -r Music this deletes the Music directory and whatever is saved in it
5. rm -f file1.txt

GeeksforGeeks, (2019)

Here is the strace for rm



I saved the output in a text file to read it easier



A screenshot of a computer

Description automatically generated

cat:

1. This command can create a file
2. It can display contents of a file
3. It can concatenate files
4. It allows a user to append a file
5. It allows a user to overwrite a file

Two main flags:

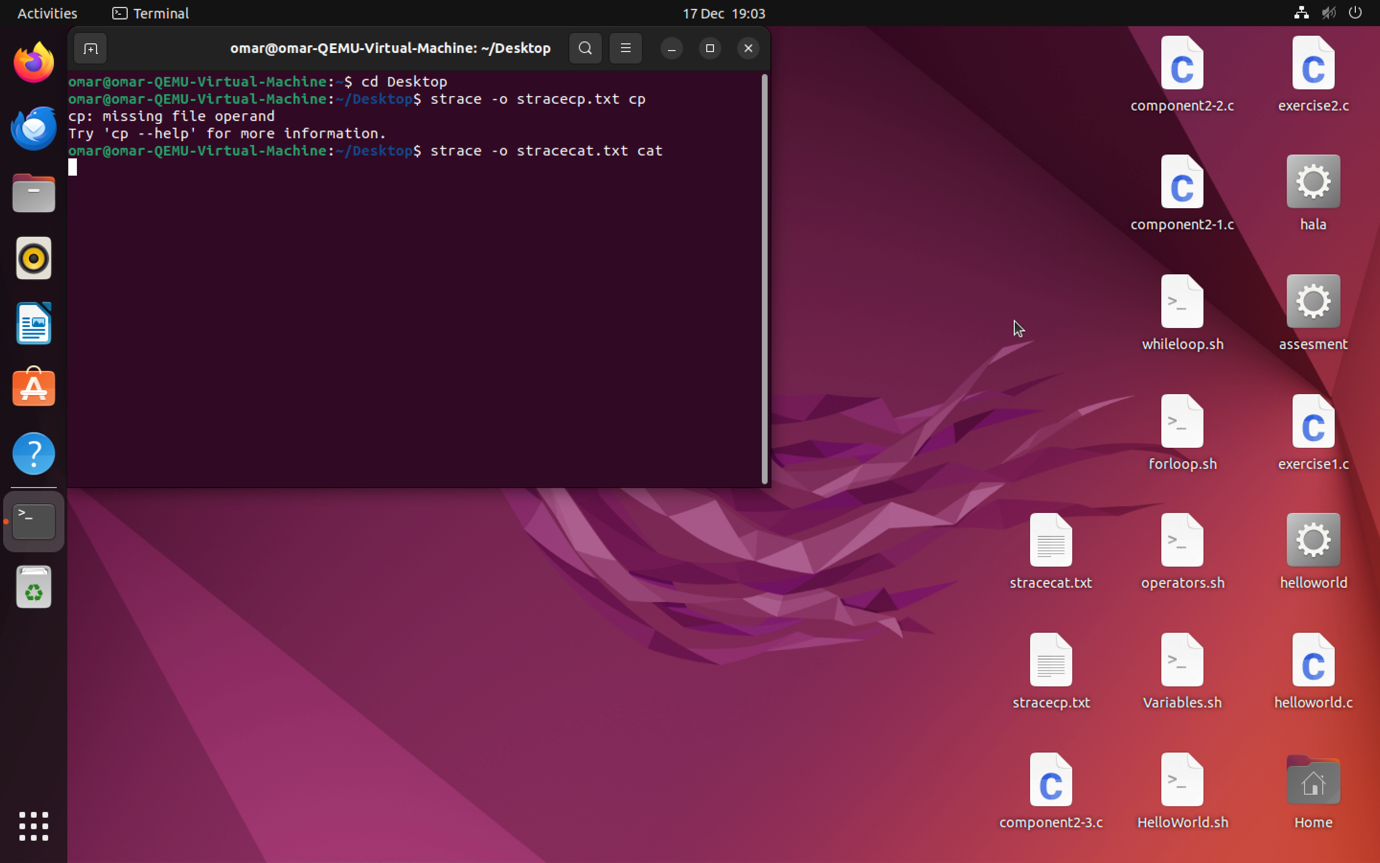
1. -v this displays nonprinting characters.
2. -n displays line numbers with each line.

cat syntax:

1. cat > file1.txt this creates a file with the name file1 and after this the user can type in the data, he/she wants into the file after that control + c to save and exit.
2. Cat file1.txt displays file1’s content.
3. Cat file1.txt file2.txt displays both files file1 and file2 content after each other (concatenates).
4. Cat file1.txt > file2.txt copies the content of file1 into file2.
5. Cat >> file1.txt appends to file1 by adding more content top the file. When the command is run the user can type in the data, he/she wants and after that control + c to save and exit.

GeeksforGeeks, (2019)

Here is the strace for cat



I saved the output in a text file to read it easier

A screenshot of a computer

Description automatically generated

Here is the table which shows what system calls were triggered by which command

|  |  |  |  |
| --- | --- | --- | --- |
| System Call Triggered | CP | RM | CAT |
| execve | Y | Y | Y |
| brk | Y | Y | Y |
| access | Y | Y | Y |
| mmap2 | Y | Y | Y |
| open | Y | Y | Y |
| read | Y | Y | Y |
| write | Y | N | Y |
| fstat64 | Y | Y | Y |
| close | Y | Y | Y |
| mprotect | Y | Y | Y |
| munmap | Y | Y | Y |
| set\_thread\_area | Y | Y | Y |
| set\_robust\_list | Y | N | N |
| rt\_sigaction | Y | Y | Y |
| rt\_sigprocmask | Y | Y | Y |
| ugetrlimit | Y | N | N |
| uname | Y | N | N |
| statfs64 | N | Y | N |
| fadvise64\_64 | Y | N | Y |
| geteuid32 | Y | Y | Y |
| stat64 | Y | Y | Y |
| ioctl | N | Y | N |
| faccessat | N | Y | N |
| unlinkat | N | Y | N |

System calls are divided into 5 main categories which are:

1. Process Control
2. File Management
3. Device Management
4. Information Maintenance
5. Communication

A diagram of a system

Description automatically generated

1. **Process Control:**

This system calls perform the task of process creation, process termination, etc.

The Linux System calls under this are fork() , exit() , exec().

1. fork() creates a new process.
2. exit() used by a program to terminate its execution.
3. exec() a new program will start executing.
4. **File Management:**

File management system calls handle file manipulation jobs like creating a file, reading, and writing, etc. The Linux System calls under this are open(), read(), write(), close().

1. open() It is the system call to open a file.
2. read() this opens a file in reading mode.
3. write() this opens the file in writing mode.
4. close() this closes the open file.
5. **Device Management :**

Device management does the job of device manipulation like reading from device buffers, writing into device buffers, etc. The Linux System calls under this is ioctl().

1. ioctl() is referred to as input and output control.
2. **Information Maintenance:**

It handles information and its transfer between the OS and the user program. In addition, OS keeps the information about all its processes and system calls are used to access this information. The System calls under this are getpid(), alarm(), sleep().

1. getpid() stands for get the process ID, it returns the process ID of the calling process.
2. alarm() it sets an alarm clock for the signal to be delivered to the calling process.
3. sleep() it suspends the execution of the currently running process for some interval of time to allow other process to run in this interval of time.
4. **Communication :**

These types of system calls are specially used for inter-process communications. Two models are used for inter-process communication and they are:

1. Message Passing(processes exchange messages with one another)
2. Shared memory(processes share memory region to communicate)

The system calls under this are pipe() , shmget() ,mmap().

1. pipe() it is used to communicate between different Linux processes. It is a inter-process communication.
2. shmget() stands for shared memory segment, it is used to access the shared memory and access the messages to communicate with the process.
3. mmap() it is used to map or unmap files or devices into memory. It is responsible for mapping the content of the file to the virtual memory space of the process.

GeeksforGeeks, (2022)

Here are some **general explanations to the system calls** mentioned in the system call table:

* **execve:** Execute program, process creation.
* **brk:** Set the end of the data segment of the calling process to a specified value.
* **access:** Check real user's permissions for a file.
* **mmap2:** Create a memory mapping.
* **open:** Open or create a file.
* **read:** Read a file.
* **write:** Write a file.
* **fstat64:** Get file status.
* **close:** Close a file.
* **mprotect:** Change memory protection (example: read-only, executable).
* **munmap:** Remove memory mapping (associate the contents of a file with a range of addresses in the system's virtual memory).
* **set\_thread\_area:** Set the thread-local storage information for a thread (thread of a multithreaded process has its own private area for storing variables).
* **set\_robust\_list:** Set the list head for robust futex list (Fast User-Space Locking).
* **rt\_sigaction:** Examine or change a signal action.
* **rt\_sigprocmask:** Examine or change blocked signals.
* **ugetrlimit:** Get resource limits.
* **uname:** Get system information.
* **statfs64:** Get file system statistics.
* **fadvise64\_64:** Provide advice about file access patterns.
* **geteuid32:** Get effective user ID.
* **stat64:** Get file status.
* **ioctl:** Perform I/O control operations.
* **faccessat:** Check file access permissions.
* **unlinkat:** Delete a name from the file system.

**Linux**

The Linux Operating System is a type of operating system that is like Unix, and it is built upon the Linux Kernel. The Linux Kernel acts as the central control centre of the operating system, overseeing how the computer interacts with its hardware and resources to ensure smooth and efficient operation. However, the Linux Kernel alone does not constitute a complete operating system. To create a comprehensive and functional system, the Linux Kernel is combined with a set of software packages and utilities, collectively referred to as Linux distributions. These distributions prepare the Linux Operating System for users to run applications and perform tasks on their computers securely and effectively. Linux distributions are available in various configurations, each tailored to cater to the specific needs and preferences of users. Its source code is open and available for everyone to explore and modify, this openness keeps Linux up to par with other operating systems as it keeps on evolving as technology emerges. It is mostly known for being efficient, meaning it can do a lot of tasks quickly, also being cost-effective, which means it doesn’t cost a lot to run and use.

**Linux design principles**

Linux is a multiuser, multitasking system with a full set of UNIX- compatible tools.

Its file system follows traditional UNIX principles, and it completely embraces the standard UNIX networking model.

The primary design objectives are speed, efficiency, and standardization.

Linux is designed to adhere to the appropriate POSIX standards, with at least two Linux distributions having obtained official POSIX certification.

The Linux programming interface aligns with SVR4 UNIX semantics rather than BSD behaviour.

The structure of Linux in the simplest diagram is this:

linux design diagram

https://www.geeksforgeeks.org/architecture-of-linux-operating-system/

1. **Kernel:**

Kernel is the main core component of Linux, It manages hardware resources, allocates virtual resources to processes, and schedules them for execution to prevent conflicts. To be exact Linux kernel is type referred to as monolithic kernel where all the concurrent processes are executed simultaneously in the kernel itself. All the processes share same memory recourses.

Main Subsystems of Linux kernel:

A diagram of a system

Description automatically generated

**Process scheduler**: Responsible for fair distribution of processing time to all running processes as a certain time.

**Memory management unit:** This subcomponent of the kernel ensures the proper allocation of memory resources among concurrently running processes.

**Virtual file system:** This subsystem provides interface to access stored data across different file system and different physical media.

1. **System Library:**

System libraries are predefined functions that allow application programs or system utilities to access kernel’s features. These libraries serve as the essential building blocks upon which all software can be constructed.

There are a lot of libraries, some are: GNU C library, libpthread (POSIX Threads), libdl (Dynamic linker), libm (Math library), librt (Realtime Library), libcrypt (Cryptographic Library), libnss (Name Service Switch Library), libstdc++ (C++ Standard Library).

1. **Shell:**

The shell can be thought of as the interface to the kernel, which hides the inner workings of the kernel's functions. Users simply input commands, and the shell takes care of executing the corresponding tasks using the kernel's functions.

There are different types of shells here is two:

A diagram of a hardware system

Description automatically generated

**Command line shell** whichexecutes the command provided by user given in the form command. It is executed in the terminal and the result is displayed in there too.

**Graphical user interface** which executes the process provided by the user in a graphical way and output is displayed in the graphical window.

1. **Hardware Layer:**

The hardware layer in Linux represents the lowest level of the operating system stack. It plays a critical role in overseeing and controlling all hardware components, including device drivers, kernel functions, memory management, CPU governance, and I/O operations. This layer abstracts the intricacies of hardware complexity, offering a software interface that ensures the proper functioning of all components.

1. **System Utility:**

System utilities are command-line tools designed to execute various tasks as requested by users, enhancing system management and administration. These utilities empower users to carry out diverse operations, including file management, system monitoring, network configuration, and user management.

GeeksforGeeks, (2023)

**comparative analysis between Windows and Linux:**

|  |  |
| --- | --- |
| Linux | Windows |
| It is not very user friendly | Very user friendly and easy to use |
| It has small peripheral hardware drivers | It has large peripheral hardware drivers |
| There 3 types of user accounts:  1.Regular 2.root 3.service account | There are 4 types of user accounts:  1.Administrator 2.standard 3.child 4.guest |
| It is more secure than any other operating system. It does not mean that Linux is 100 percent secure, it has some malware for it but is less vulnerable than any other operating system. So, it does not require any anti-virus software. | Poor security because it is much more prone to viruses and malware in comparison, so it requires anti-virus software. |
| It is fast and easy to install from the web. It can also install on any hardware even on old computer systems. | It needs more advanced hardware to be installed and run because the GUI uses more graphics so needs a good GPU. |
| Uses a monolithic kernel. | Uses a hybrid kernel. |
| Supports multiple file systems like ext3, ext4, Btrfs, XFS, etc. | Primarily uses NTFS, with support for FAT32 and exFAT. |
| Highly customisable, from the kernel up to the desktop environment. | Highly customisable, from the kernel up to the desktop environment. |
| Mostly free; some distributions come with optional paid support. | Commercial product; requires purchasing a license. Free upgrades within product versions. |
| Preferred for servers, programming, and as an operating system for IT professionals and developers. | Dominant in personal computing, enterprise environments, and gaming. |
| Updates are handled through a centralized package manager, with more control over what gets updated and when. | Updates are managed by Microsoft, with periodic mandatory updates for security and system features. |
| Known for stability and efficiency, particularly in server environments. It handles system resources efficiently. | Improved performance over time but can be resource-heavy. It’s optimized for personal computing and gaming. |

**Windows**

Windows is an operating system that utilises graphical user interface and is less technical than Linux. This operating system is provided by Microsoft so requires you to buy a license to run it legally. Windows shares the same three commands that Linux have and here they are in a table displayed opposite to their counterparts in Linux.

|  |  |
| --- | --- |
| Windows command | Equivalent Linux command |
| copy/xcopy | cp |
| del | rm |
| type | cat |

They also have system calls that run when the command is run and some of them have an equivalent in Linux here is a table displaying that.

|  |  |
| --- | --- |
| Windows system call | Equivalent Linux system call |
| CreateProcess | execve |
| GetFileAttributes | access |
| CreateFileMapping | mmap2 |
| CreateFile | open |
| ReadFile | read |
| WriteFile | write |
| CloseHandle | close |
| VirtualFree | munmap |
| GetVersionEx | unname |
| DeleteFile | unlinkat |

Here are some **general explanations to the windows system calls** mentioned in the table above:

* **CreateProcess:** Used to create a new process and its primary thread. This process runs the specified executable.
* **GetFileAttributes:** Retrieves attributes for a specified file or directory, such as whether it's read-only, hidden, a system file, etc.
* **CreateFileMapping:** Creates or opens a file mapping object for a specified file. This is used for interprocess communication or to map a file to memory.
* **CreateFile:** Opens a file, device, directory, or volume. This function can also create a file or device if it doesn't exist.
* **ReadFile:** Reads data from a file, input/output (I/O) device, or communications resource.
* **WriteFile:** Writes data to a file or an I/O device.
* **CloseHandle:** Closes an open object handle, which could be a handle to a file, file mapping, process, thread, etc.
* **VirtualFree:** Releases or decommits a region of pages within the virtual address space of a calling process.
* **GetVersionEx:** Retrieves information about the version of the operating system that is currently running.
* **DeleteFile:** Deletes an existing file.

**References:**

GeeksforGeeks, (2022) Linux System Call in Detail [online]. Available from: <https://www.geeksforgeeks.org/linux-system-call-in-detail/>

Computer Hope, (2021). Linux Cat Command Help and Examples [online]. Available from: <https://www.computerhope.com/unix/ucat.htm>

GeeksforGeeks, (2023). Introduction to Linux Operating System [online]. Available from: <https://www.geeksforgeeks.org/introduction-to-linux-operating-system/>

GeeksforGeeks, (2019). Linux Tutorials | cat - A versatile command | GeeksforGeeks [video]. YouTube. Available from: https://youtu.be/exj5WMUJ11g?si=OscHs\_lKS5chRiw-

GeeksforGeeks, (2019). Linux Tutorials | cp - Copying a File | GeeksforGeeks [video]. YouTube. Available from: https://youtu.be/ngx4G10XY2s?si=6I2sXTDCcN1suiGt

GeeksforGeeks, (2019). Linux Tutorials | rm - Removing a file | GeeksforGeeks [video]. YouTube. Available from: https://youtu.be/gcE5QvggzL4?si=w0MFJ\_wvLdv5qKg2