The Linux Process Manager

Maya Hussein, Nada Badawi, & Farah Awad
Department of Computer Science & Engineering
The American University in Cairo

CONTENTS

The Linux Process Manager

Keywords—component, formatting, style, styling, insert

I. Introduction

This survey aims to demonstrate the features and capabilities of existing tools in managing the system processes. The Linux Process Manager has some limitations and desirable features that we would deploy in our project. The survey explores the existing tools available to manage processes in a Linux environment.

II. Background

A process is a program in execution. When a program is launched, one or more processes are created in memory to execute its instructions. A process has an allocated set of system resources including input/output devices, file handlers, and networking connections. The operating system is responsible for managing processes, including creating processes & scheduling processes, and allocating the system resources. The process manager is the component that operates these functions. The manager processes for errors, terminate the processes that have stopped responding, and provides synchronization. [1]

III. Discussion

A. *Process System Administration Common Tasks*

A process manager's main goal is to run different processes efficiently while allocating the suitable resources; such as CPU time, memory, and I/O bandwidth. To achieve its goal, there are some common tasks that a process manager is expected to perform. Firstly, it keeps track of the processes status, spotting those that are using too many system resources or have slowed down, and taking corrective action when problems are found. This corrective action may be launching processes or halting them when they're finished. This is achieved through setting the priority of processes to ensure that the critical ones receive the required system resources first.

Another crucial task that a process manager carries out is defining the scheduling policy to control how CPU time is distributed across tasks. The CPU time is not the only process quota that needs configuring, but also the memory usage and disk space. Process managers also set-up inter-process communication mechanisms (IPC), such as pipes, sockets, or message queues so that processes can communicate with one another. [2]

B. *Existing Tools*

The following are the most popular Linux Process Managers. They are not necessarily the top ten since some of them were used for old computers. The list consists of 5 command-line based tools; the other 5 are graphical-user based.

1) Top: - The top tool, which stands for Table Of Processes, is a very efficient process manager. The system resources can be easily monitored using this tool. Although, it is a command line tool, the interface is easy to use and understand. The data that the tool can display is the CPU usage, the SWAP usage, the RAM resource usage, and can keep track of the total processes currently running. Something that is quite helpful in this tool is that it displays the state of each process, whether it is a parent, child, orphan, zombie, or a daemon. However, a disadvantage of this tool is that managing a process requires the process ID. For instance, to prioritize a task, we need to know its PID to be able to execute this command [3].

2) Glances: - This tool is special in customizing the manager's functionality, which adds flexibility. The tool is also command-line based (CLI) but can display a huge amount of data about a single process. Another feature that makes this tool special is the ability to monitor the processes of a computer, manage, and export their data from any device [3].

3) Htop: - This tool also allows many information about a single process to be displayed, but it is also user friendly despite being CLI-based. An exciting feature in this tool is that it can filter and manage processes. However, it does not provide any information about the SWAP memory, which is the virtual memory that programs need if the physical memory is insufficient [1].

4) Ps: - This tool is already in Linux; therefore, no further steps should be taken to run it, such as installing it. The tool has multiple features that might be handy, such as sorting the processes according to their CPU consumption. Moreover, it can show all threads of a particular process [6].

5) Pstree: - This is merely an extension of the Ps tool by which the data representation is different [1]. As the name suggests, the displayed data is in the form of a tree where the root is the systemd.

6) GNOME: - This is a GUI-based (Graphical User Interface) tool that can view and manage processes along with their data, such as CPU consumption and memory consumption. Moreover, it allows viewing the hard drive space through the tab 'File Systems' tab. Its simplicity is what makes it helpful and useful. The tool is already available in Linux distributions, so installation is not required [5].

7) Conky: - This tool provides extra data that is not provided by most of the rest of the process managers. This includes the system's temperature, disk usage, and network stream (both upstream and downstream). It still manages processes and displays them with their necessary information [6].

8) XFCE4: - This tool, along with its GUI, helps the user to track and manage the processes. The user can easily change the priority of the process or can change its state by killing it. The tool still allows to monitor the system resources [1].

9) Stacer: - The Stacer tool is considered an all-in-one process management tool. It is the perfect fit for those who are still not experienced when using linux-based systems. It has the history of resources usage. In addition, it can optimize the system and clean it to yield the best performance [5].

10)LXTask: - This tool may not be the best for newer systems. It can only view the roots of currently running processes on LXDE/LXQt desktop environments. This desktop environment is considered old were used for computers with weak hardware and performance [6].

C. Capabilities

Prior to POSIX (Portable Operating System Interface), a set of IEEE standards used to maintain compatibility among operating systems, traditional protection domains allowed files to be accessed by identification numbers, and kernel access were privileged to only super users. A process had access to all privileges or none. However, with POSIX, root privileges were partitioned into discrete capabilities, all, some, or none allowing for a "minimum necessary" set of grants. Such policy allocates a subset of capabilities for a process to carry out its current task as long as they are needed. [4]

For every process, three groups of capabilities exist, where every set coincides with the previlages currently allocated for usage. These sets are encoded in 32-bit bitmaps in the task struct. The inherited set specifies the capabilities a process can aquire after execution. Such capabilities can be temporary if needed.

Capabilities are defined by their mnemonic name, defined as a constant starting with CAP_, and can be found in "linux/capability.h" library. They can be represented in the following data structures and functions based on their managers.

1) File System Capabilities: This set grants capabilities related to accessing filing systems. Linux file system managers allow users to create, delete, and set permissions on files and directories. It also allows resizing existing file systems, and changing the properties of file systems. [4]

Here is a list of its capabilities:

- CAP_CHOWN:- Changes user & group ownership of a file
- CAP_DAC_OVERRIDE:- overrides all access controls on files.
- CAP_FOWNER:- overrides all restrictions on file operations based on user ID.
- CAP_DAC_READ_SEARCH:- overrides read & search restrictions on directories and files.

In the linux/capability.h library, a bitmask, *104 define CAP_FS_MASK 0x1f*, contains bits corresponding to all the capabilities in this set.

2) Process Management Capabilities : This group allows users to create, manage, and terminate processes. This includes starting new processes, stopping existing processes, and monitoring process status. [4]

1) CAP_KILL:- override restrictions on signal transmission. (Signals are sent to any other process)
2) CAP_SETGID:- grants use of setgid(), sets group ID for process, & setgroups(),sets supplementary groups for a process, system services.
3) CAP_SETUID:- grants use of setuid(), sets user id of current process.
4) CAP_SETPCAP:- remove or transfer capability from one process to another

3) Process Management Capabilities - Set Two: This another set of capabilities that is related to the process management. [4]

1) CAP_SYS_PTRACE:- grants use of ptrace() on any process. Contrary, a traceable process has to be the child of the tracing process.
2) CAP_SYS_PACCT:- grants process accounting configuration
3) CAP_SYS_ADMIN:- grants general system administration
4) CAP_SYS_BOOT:- grants use of reboot
5) CAP_SYS_RESOURCE:- grants and overrides quota and resource limits
6) CAP_SYS_TIME:- grants editing of clock (system & realtime)

4) Networking Capabilities : This set is associated with networking; it provides tools to manage network connections, including establishing and terminating connections, monitoring connection status, and troubleshooting network issues, and to monitor network traffic, including tools for capturing and analyzing network packets, monitoring network usage and performance, and identifying network bottlenecks. [4]

1) CAP_LINUX_IMMUTABLE:- allows S_IMMUTABLE & S_APPEND to modify files
2) CAP_NET_BIND_SERVICE:- allows for TCPUDP sockets below 1024 to be bounded
3) CAP_NET_RAW:- grants use of SOCK_RAW sockets
4) CAP_NET_BROADCAST:- allows for network broadcasting
5) CAP_NET_ADMIN:- grants general network admin

D. Capabilities to be avoided

In order to protect the security and stability of the system, a process manager should avoid using certain features. For starters, a process manager should not grant non-privileged users access to control system operations so that it prevents unauthorized users from interfering with system operations, only users with administrator credentials should be able to handle system processes. To elaborate, Process administrators should limit access to system resources, including CPU time, memory, and disc space, to make sure that system services and user applications may function properly and without interruption. Limitation of access can be achieved by employing strong authentication and authorization measures to stop unauthorized users from entering the system or taking over its operations. Another unfavorable process manager capability allowing arbitrary code execution since it might result in security flaws and jeopardize the stability of the system. It is also advisable to limit the privileges of processes so that they only have access to the system resources they need to do their functionalities. Accordingly, processes are prevented from running with excessive access, interfering with the operation of other system processes, and from using system resources without authorization. [7]

E. Lacking Capabilities

While Linux process manager provides many unique features, it lacks some advanced capabilities that are worth noting. To elaborate, basic process information can be displayed by Linux process managers, however complex visualizations of process hierarchies and linkages might not be available. Furthermore, Linux process managers do not offer control over certain process parameters as CPU affinity and I/O priority, yet they are capable of starting, stopping and managing processes. Another functionality that the process manager lacks is deep system monitoring, which is providing comprehensive system-level monitoring for network traffic, file system activity, and other system-level metrics. With regards to compatibility, Linux process managers may not be compatible with other types of operating systems or architectures, since it is made to particularly function with Linux-based operating systems. Linux also does not provide native support for process virtualization technologies such as containers or virtual machines. [3]

F. Desirable Capabilities

Depending on the particular use case and context, different features in process managers may be useful; nevertheless, some common qualities that are typically desirable include real-time process monitoring. This enables the provision of real-time information on the state of processes and resource utilization, enabling the early identification and resolution of problems. Generally, conducting thorough process tracing and debugging is a highly required feature in any process manager. One of the methods that reduces errors is automating procedures and operations using scripts or other programming tools. A good process manager should also allocate resources to processes in a way that enhances system performance and throughput. Furthermore, it has the ability to schedule and prioritize processes according to their significance, resource needs, and other considerations in order to improve system performance and cut down on wait times. Needless to say, it is of paramount importance to implement security policies and access restrictions, as well as to stop unwanted access to system resources or processes. The security issue becomes rather tricky when virtualization takes place. To elaborate, a process manager should be able to separate environments to be managed for running programmes, a process known as virtualization, enhancing security and lowering the possibility of process conflicts. Additionally, it is preferred to have a process manager that has the capacity to work with other monitoring and management tools as well as other system elements like networking, storage, and security systems. [6]

G. Proposed Functional Requirements

Below is a list of the proposed functional requirements. The process manager will display the following:

1) Process:- all current processes
2) Process ID
3) Process State:- state of the process in manager, i.e. Zombie, Daemon, Orphan, Child, Parent
4) CPU Utilization
5) CPU Execution Time
6) Memory Management:- display the amount of space taken up in the storage
7) Display file names opened by the process
8) User ID_Group ID

Below is also a list of functionalities that will be implemented in the process manager.

1) Process Termination:- able to terminate the process
2) Process Scheduling
3) Recover loss of data after termination of the process
4) Sort process based on the process type i.e. user or system Filter the processes based on CPU Measurements i.e. CPU Utilization, CPU Time, memory usage, etc:.
5) Error-Handling:- The process manager has to be able to handle breaches in privileges

References

[1] Linux scheduler (no date) Linux Scheduler - The Linux Kernel documentation. Available at: https://www.kernel.org/doc/html/latest/scheduler/index.html (Accessed: February 19, 2023).

[2] IBM developer. Available at: https://developer.ibm.com/tutorials/l-linux-process-management/ (Accessed: February 19, 2023).

[3] Strang, J., Todino, G. and Peek, J. (no date) Learning the unix operating system, 5th Edition, O'Reilly Online Learning. O'Reilly Media, Inc. Available at: https://learning.oreilly.com/library/view/learning-the-unix/0596002610/ (Accessed: February 19, 2023).

[4] O'Gorman, J. (2003) The linux process manager: The internals of scheduling, interrupts, and signals. Chichester, England: Wiley.

[5] Nemeth, E., Hein, T.R. and Snyder, G. (2009) Linux Administration handbook. Upper Saddle River (New Jersey): Pearson Education.

[6] Shotts, W.E. (2019) The Linux Command Line: A complete introduction. San Francisco, CA: No Starch Press.

[7] Niazi, R. (2022) Linux Process Management: The ultimate guide, MUO. Available at: https://www.makeuseof.com/linux-process-management/ (Accessed: February 19, 2023).

[8] Rathore, D. (2023) 15 Best Linux Task Managers, Dunebook. Available at: https://www.dunebook.com/best-linux-task-managers/ (Accessed: February 19, 2023).