JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A STUDY ON CIRCUMSTANCES OF FORK ( ) AND EXEC ( ) PROCESS MODEL IN LINUX SYSTEM

Mrs.Sowmya K N,
Assistant Professor,
Department of Computer Science,
St.Joseph's First Grade College, Jayalakshmipuram Mysore-12

Abstract: This article gives more information about the advanced concepts of operating system. We can say, operating system is an interface between users (people who are using the computer) and the hardware (Physical components of an electronic system). In this article we will reach a brief explanation of the fork () and exec () system call and how it works. Operating system has two modes: Kernel mode and User mode. Kernel mode is convenient for different reasons. Anybody can write kernel code because Linux's code is free. Kernel mode is used to access physical components and also manages application software and hardware interaction. Kernel mode directly interact with the hardware. When program running under user mode, it needs hardware interaction. User mode does not have direct access to system resources.

Keywords: Kernel, Linux, Process, fork (), exec ()

Introduction: Program is a specific set of instructions and data can be executed once, many times or simultaneously is called programs and this programs are executed by a CPU on its instructions called process. The process is the basic context of said program so those processes can be run many programs which are operating system processes(system code) and the remains of which are user processes(user code)

The computer system has two modules 1) kernel modules 2) user modules. In kernel module a program can directly access memory and hardware resources. When user modules, programs may not directly access hardware as well as memory resources, so kernel module is more privileged than user module. The Linux System uses a variety of system calls like fork ( ), exec ( ), wait () and exit ().

1.**Process model of the fork (): UNIX** process management contains two operations.

a.The creation of new process is called fork () system calls in Linux, whose primary reason is to create a new process, this is done by copying or clone of the parent process. It recreates itself, the parent process is the existing process and the child process is the derived process. Until the child process runs completely, the parent process is suspended.

Ex:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main()

{
if (fork() == 0)
{
printf("Hello from Child Process\n");
}
else{
printf("Hello from Parent Process\n");
}
}
```

When running the above program, we get the output as below:

```
Hello from Child Process
Hello from Parent Process
```

The fork () system call returns two results: 0 or PID. If fork () returns 0 values, it means it is a child process. In an another part, in the parent process fork () returns the PID of the child process.

Since, an if statement is used to get only one result needs to be displayed but in the result we will get both the results.

How both the results are displaying? In such cases, the fork () system call will create a new copy of the running process i.e child process with the same code. So in the parent process the if condition become false and result after else will get displayed. But in the child process the if condition will become true and the result after it will get displayed. This is what we will get two results from an if else statement.

Note: fork () is a command it allows to copy itself. The fork () construct a child process equal to the parent's process.

2.**Process model of the exec ():** The system call is used to construct the processes. When the exec () system call is used, the currently running process is terminated and it is replaced with the newly formed process. In another way, only the new process continue after calling exec () function the parent process's address memory, data segment and text segment with the child process.

Note: exec () is a command that constructs a new process by exchanging the existing process. In this child address memory replaced with parent address memory.

Under UNIX system must maintain to track the context of a single execution of a single program, but in Linux can break down this context into a number of specific sections. This can be classed into three.

a.Process identity: The process identity includes:

- Process ID: Each process has a separate identifier. When an application makes a system call to wait, modify or signal for another program then it is used to specify processes to the operating system. It is unchangeable and uniquely identifies that process until termination.
- Credentials: Every process must have connected with userID and one or more user group that tells the rights of a process to access system resources and files. The process group and identifiers can be changed. If the process wants to start a new group. Its credentials can be changed.
- Personality: Process personalities not used in the UNIX systems but in the Linux each process has an associated identities that can modify a bit to the system calls.

b.Process Environment: process environment is derived from their parent and designed into two vectors. They are argument vector and environment vector.

Argument vector is grouped the command line argument used for calling the running program. Environment vector is not in the kernel memory but stored in the user mode address

When a new process is created, the argument and environment cannot be changed. The child process inherits the environment from its parent possesses. The new environment will set up when a new program is called. The new program get the information from the environment. The kernel forward the environment for the next program and replace the current environment.

Process Context: Process identity and environment properties are arranged when a process is created and cannot be changed up to that process completion. A process can select to change some features of its identity if it is required or can be altered its environment. In distinction, process context running the program at any one time, it can changes running the program continuously.

Parts of process context:

Scheduling context: scheduling context is the most important part of the process context. Scheduler needs to suspend the information and restart the process. This information contains saved files of all the process registers. In scheduling context floating point do not use because it stores separately and restored only when needed. Scheduling context also contains information about scheduling priority and any outstanding signals waiting to be bring into the process.

Accounting: each process is consuming the currently resources which are maintained by the kernel and it keeps the information about the total resources consumed by the process in its lifetime.

Conclusion: fork () system call, that we can use to create new processes in Linux. The fork system calls is usually used within a running process

Running a new program does not require a new process to be created first. Any process can call exec () at any time. The currently running program may immediately terminated and the new program starts executing in the context of the existing process. It has great simplicity, no need to specify every detail of the environment of the new program in the system call.

The parent process and child process share the same code and also share the same data segment also. Only when one of the process tries to change the copied data .in clone () command allows the parent and child process to share more resources by that time they can have same PID and differ by their stack segments and processor register value.

References:

[1] Lee, E.A., 2008, May. Cyber physical systems: Design challenges. In *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC)* (pp. 363-369). IEEE

[2] Wolbers, Maarten HJ, Paul M. De Graaf, and Wout C. Ultee. "Trends in the occupational returns to educational credentials in the Dutch labor market: changes in structures and in the association?." *Acta sociologica* 44.1 (2001): 5-19.

[3] Sekar, R., Bendre, M., Dhurjati, D., & Bollineni, P. (2000, May). A fast automaton-based method for detecting anomalous program behaviors. In *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001* (pp. 144-155). IEEE.