Scientific Journal of Impact Factor (SJIF): 4.14
e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 3, Issue 2, February -2016

# Retrieving a Process Control Block (PCB) of Each Process Executing in Client System using RPC

Anand Pandya[1] , Nirali Thakkar[2], Pratik Soni[3],Pritesh Pandey[4], Sunit Parmar[5]

[1]*Assistant ProfessorAssistant Professor, Information Technology Department, A.D.Patel Institute of Technology, New VallabhVidyanagar, Gujarat, India*

[2,3,4,5]*Assistant Professor Assistant Professor, Computer Science and Engineering Department, Madhuben and Bhanubhai Patel Women Institute of Engineering , New VallabhVidyanagar, Gujarat, India*

***Abstract :*** *In distributed systems, the clients having too many process running in its system. Each time server has to check the states of its currently running process. Using RPC, Server can retrieve the status of all the processes and its status (running, sleeping, etc.) of all the clients. After getting the information of all the this information, server can easily distribute the work among all the clients .Client to server remote procedure calls are used when you want to send client data to the server. This paper proposes that the all the clients sends the status of currently running process to the server and server keeps track of all that information coming from all the clients.*

***Keywords:****Client stub, Distributed system, Process Control Block, proc file system, RPC, Server stub.*

## I. INTRODUCTION

RPC is an inter-process communication that allows a computer program to cause a subroutine or procedure to execute in another address space without the programmer explicitly coding the details for this remote interracial Call is started by client. It sends a request to server to execute remote procedure with appropriate parameters. While the server is processing the call, the client is blocked (it waits until the server has finished processing before resuming execution), unless the client sends an asynchronous request to the server.

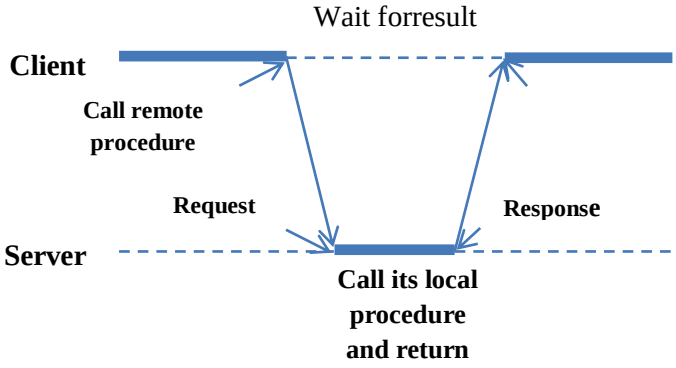


Fig. 1 Client – Server Architecture in RPC

The above the architecture shows the Sequences of events during the Remote Procedure. The client calls the client stub. The call is a local procedure call, with parameters pushed on to the stack in the normal way. The client stub packs the parameters into a message and makes a system call to send the message. Packing the parameters is called marshaling. The client's local operating system sends the message from the client machine to the server machine. The server stub unpacks the parameters from the message. Unpacking the parameters is called unmarshalling. Finally, the server stub calls the server procedure. The reply traces the same steps in the reverse direction. [1]

A **stub** in distributed computing is a piece of code used for converting parameters passed during a Remote Procedure Call (RPC). The client and server use different address spaces, so conversion of parameters used in a function call has to be performed; otherwise the values of those parameters could not be used, because of pointers to the computer's memory pointing to different data on each machine.

This paper proposes the combination of proc system call with RPC to retrieve the information of client by server. Client sends this information using message passing.

 

The procfilesystem is a pseudo-filesystem which provides an interface to kernel data structures. It is commonly mounted at /proc. Most of it is read-only, but some files allow kernel variables to be changed.

***/proc/[pid]***

There is a numerical subdirectory for each running process; the subdirectory is named by the process ID. Each such subdirectory contains the pseudo-files and directories. proc files are created when system get started. The directories in the /proc directory created automatically when process is created and deleted automatically when process is terminated. /proc is haviely used in the Linux systems. Many utilities on a morden Linux distribution such as psetc get their information from /proc file systems. [2][4]

The rest of this paper is organized as follow. This section discusses the RPC and /proc file system. Section 2 describes the proposed approach.

## II. PROPOSED APPROACH

This paper proposes the combination of RPC and /proc pseudo file system to retrieve PCB (Process Control Block) of currently running process. The /proc file system contains a directory entry for each process running on the GNU/Linux system. The name of each directory is the process ID of the corresponding process 1. These directories appear and disappear dynamically as processes start and terminate on the system. Each directory contains several entries providing access to information about the running process. Each process contains following entries:

- cmdline contains the argument list for the process.
- cwd is a symbolic link that points to the current working directory of the process
- environ contains the process's environment
- exe is a symbolic link that points to the executable image running in the process.
- fd is a subdirectory that contains entries for the file descriptors opened by the process.
- maps displays information about files mapped into the process's address.
- root is a symbolic link to the root directory for this process. Usually, this is a symbolic link to /, the system root directory. This can be changed by a process using chroot.

The following section shows sequence of massage passing using RPC. Fig. 1 shows the pictorial representation of the same.

### A. *Client Side*

Client sends a request to server to call remote method type_of_info(). This method indicates client that which type of information server require. (1. PCB of each process, 2.Currently avilablable users, 3.CPU information and configuration). Based on the choice of server client execute following local procedures:

**Process_Table():** This subroutine uses /proc to retrieve process' information and stores into the buffer (The output of this method is identical to "ps –el" command).

**Currently_Logedin()**: This subroutine finds out the currently logged in users and store it into the buffer. (The output of this method is identical to" who" command).

**Cpu_Info():** This subroutine gives all the information about the client and keep track of these information into buffer.

### B. *Server Side*

Sequence of the operation done by the server is shown below:

- First, server receives the request of calling type_of_info() from the client and returns specific value based on client gives its data to server.
- According the return value client call its local procedure and again send a request to call remote procedure get_data (char *buffer). When server accepts the client's request and call get_data method it receives all the information about the client.
- get_data() method receives the information of client and displed it to sever side.

The scenario shown in the fig 2 is only for one client and one server.

 

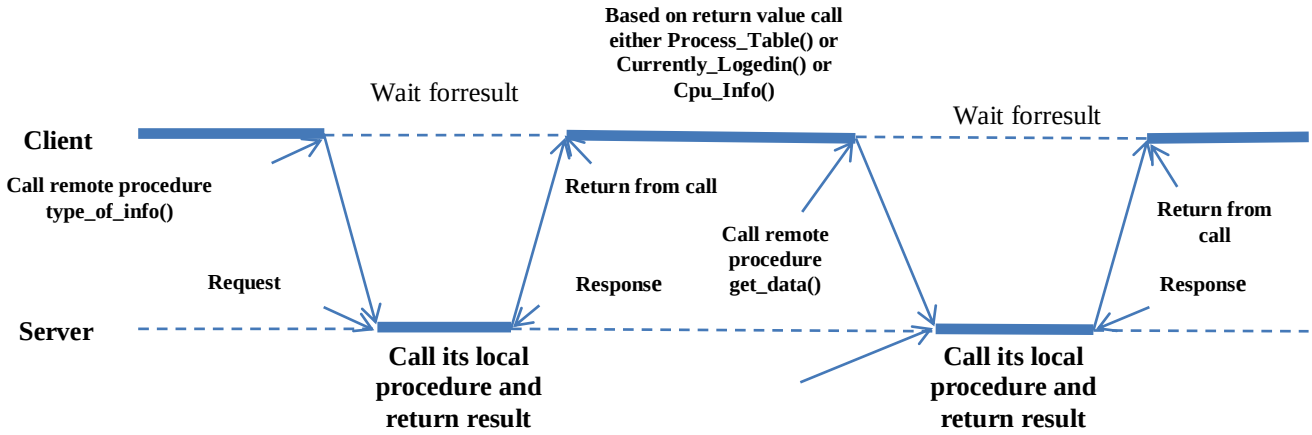


Fig. 2Proposed Scenario forsingle server and single client using RPC

## III. MULTIPLE CLIENTS AND SINGLE SERVER ARCHITECTURE

In the above section, single client and server communicate with each other. First, client sends the request and based on the request the server responds to the client and sends information type to client. Then client sends required data to the server. This paper proposes the multiple client and single server architecture which can be used in distributed system as well as in parallel processing. This is implemented on homogeneous system. The following fig shows the multiple clients and single server architecture. All clientshaving Process_Table(), Currently_Logedin() andCpu_Info() procedures and based on server choice they send their information to the server.

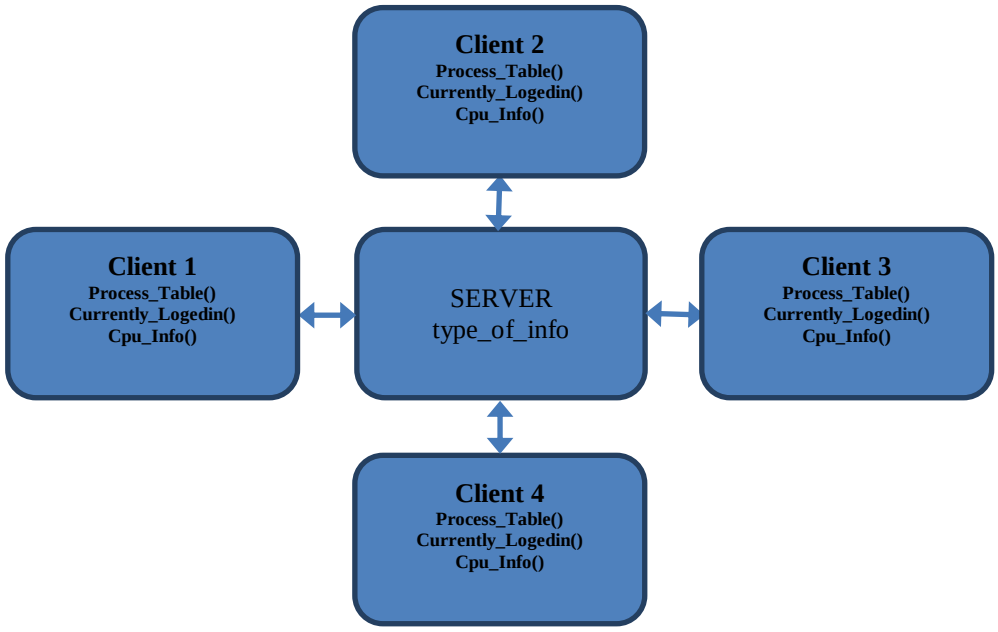


Fig. 3Proposed Scenario forsingle server and multipleclient using RPC

 

## IV. CONCLUSION

This paper proposes the use of Remote Procedure Call to communicate client and server. a remote procedure call (RPC) is client/server system in which a computer program causes a subroutine or procedure to execute in another address space without the programmer explicitly coding the details for this remote interaction. Using RPC, Server can save all the information about the client like process information, hardware detail etc. This will use in distributed system and in parallel processing.

## REFERENCES

[1] Andrew S. Tanenbaum,Maarten van Steen "Distributed Systems: Principles and Paradigms", *2nd* ed. Pearson.
[2] Manual pages of /proc file system.
[3] Meeta Gandhi, TilakShetty, Rajiv Shah,"The 'C' Odyssey Unix", BPB Publications.
[4] Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman,"Linux Device Drivers", 3rd ed., O'REILLY.
[5] Michael Kerrisk,"The Linux Programming Interface", 1st ed. ISBN-10: 1593272200.

 