BVRIT HYDERABAD College of Engineering for Women
(AUTONOMOUS)
(Approved by AICTE | Affiliated to JNTUH | Accredited by NAAC with Grade 'A' &NBA for CSE, ECE, EEE, & IT)
Bachupally, Hyderabad-500090
Department of CSE(Artificial Intelligence and Machine Learning)

Subject name: Operating System

Project name: System Call Tracer

Submitted by:

Roll Number	Name
22WH1A6619	Lakshmi Indu K

Problem Statement:

Our task involves developing a system call tracer program to monitor and analyze system calls in Linux environments. This tool aids in debugging, security analysis, and performance optimization by providing insights into process behavior.

ABSTRACT:

Introduction:

The System Call Tracer project leverages advanced scripting techniques to automate system call monitoring. By employing functions, loops, and conditional statements, we enhance command-line automation, enabling efficient system call tracing.

Implementation Details:

Starting with basic shell scripting, we integrate advanced concepts for robust tracing. Functions ensure code modularity and reusability, while loops automate repetitive tasks. Conditional statements facilitate decision-making. Practical examples demonstrate the tool's evolution, from simple scripts to sophisticated tracing mechanisms, showcasing its versatility in debugging and analyzing system behavior.

Performance Analysis:

Advanced scripting enhances tracing performance by reducing redundancy and automating tasks. Modularization improves maintainability, while loops optimize efficiency. Conditional statements enable adaptive tracing behavior, promoting streamlined code organization and error reduction. Overall, our project illustrates how scripting elevates system call tracing, empowering users to analyze and optimize system performance effectively.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ptrace.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <sys/reg.h>

int main() {
   pid_t child;
   long orig_rax;
   int status;

   child = fork();

   if (child == 0) {
      ptrace(PTRACE_TRACEME, 0, NULL, NULL);
      execl("/bin/ls", "ls", NULL);
   } else {
      wait(&status);

      while (WIFSTOPPED(status)) {
         orig_rax = ptrace(PTRACE_PEEKUSER, child, 8 * ORIG_RAX, NULL);
         printf("System call number: %ld\n", orig_rax);
         ptrace(PTRACE_SYSCALL, child, NULL, NULL);
         wait(&status);
      }
   }

   return 0;
}
```

Output:

```
System call number: 59
System call number: 12
System call number: 12
System call number: 158
System call number: 158
System call number: 21
System call number: 21
System call number: 257
System call number: 257
System call number: 5
System call number: 5
System call number: 9
System call number: 9
System call number: 3
System call number: 3
```