4ITRC2 Operating System Lab

Lab Assignment 4

Q: Comprehensive study of different categories of Linux system calls, categorized as

1. Process Management System calls
   fork(), exec(), wait(), exit().

   fork()
   Purpose: Creates a new child process.

   Syntax: pid_t pid = fork();

   Example:

```
#include <stdio.h>
#include <unistd.h>
int main() {
    pid_t pid = fork();
    if (pid == 0)
        printf("This is the child process\n");
    else
        printf("This is the parent process\n");
    return 0;
}
```


   exec()
   Purpose: Replaces the current process image with a new one.

   Syntax: execl(path, arg0, arg1, ..., NULL);

   Example:

```
#include <unistd.h>
int main() {
    execl("/bin/ls", "ls", "-l", NULL);
    return 0;
}
```

   wait()
   Purpose: Waits for the child process to terminate.

   Syntax: pid_t wait(int *status);

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

```
#include <stdio.h>

int main() {
    pid_t pid = fork();
    if (pid == 0)
        execl("/bin/ls", "ls", NULL);
    else
        wait(NULL);
    return 0;
}
```

exit()

Purpose: Terminates the process.

Syntax: void exit(int status);

Example:

```
#include <stdlib.h>

int main() {
    exit(0);
}
```


2. File Management System calls

open(), read(), write(), close().

Purpose: Perform file operations (open, read from, write to, and close files).

Example:

```
#include <fcntl.h>
#include <unistd.h>

int main() {
    int fd = open("example.txt", O_CREAT | O_WRONLY, 0644);
    write(fd, "Hello, file!\n", 13);
    close(fd);
    return 0;
}
```

3. Device Management System calls
   read(), write(), ioctl(), select().

ioctl()

Purpose: Manipulate underlying device parameters.

Example:

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/ioctl.h>

int main() {
    int fd = open("/dev/tty", O_RDONLY);
    int result;
    ioctl(fd, TIOCMGET, &result);
    close(fd);
    return 0;
}
```

select()
Purpose: Monitor multiple file descriptors.

Example:

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

int main() {
    fd_set fds;
    FD_ZERO(&fds);
    FD_SET(0, &fds);
    select(1, &fds, NULL, NULL, NULL);
    printf("Input detected.\n");
    return 0;
}
```


4. Network Management System calls
socket(), connect(), send(), recv().

socket(), connect(), send(), recv()

- **Purpose**: Create and manage socket-based communication.
Example:

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>

int main() {
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in server = {AF_INET, htons(80),
inet_addr("93.184.216.34")}; // example.com
```

```
    connect(sock, (struct sockaddr *)&server, sizeof(server));
    send(sock, "GET / HTTP/1.1\r\nHost: example.com\r\n\r\n", 39,
0);
    char buffer[1024];
    recv(sock, buffer, sizeof(buffer), 0);
    printf("Response: %s\n", buffer);
    close(sock);
    return 0;
}
```

5. System Information Management System calls

getpid(), getuid(), gethostname(), sysinfo()

- **Purpose: Get system or process-related information.**

Example:

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



```
int main() {
    printf("PID: %d\n", getpid());
    printf("UID: %d\n", getuid());

    char hostname[1024];
    gethostname(hostname, sizeof(hostname));
    printf("Hostname: %s\n", hostname);

    struct sysinfo info;
    sysinfo(&info);
    printf("Uptime: %ld seconds\n", info.uptime);
    return 0;
}
```