Operating System Lab

(4ITRC2)

IT IV Semester

Submitted by

Himanshu Priyadarshi Ahirwar

23I4026

IT-A

Submitted to

Jasneet Kaur

Department of Information Technology

Institute of Engineering and Technology

Devi Ahilya Vishwavidhyalaya, Indore (M.P.) India

(www.iet.dauniv.ac.in**)**

Session Jan-May, 2025

Study of System Calls

1.Process Management System calls:-

- **fork()**

  Purpose: Creates a **new process** by duplicating the current process.

  - The new process is called the **child process**.
  - It receives a unique process ID (PID).

  Syntax: pid_t pid = fork();

  Example:

  C

```
if (fork() == 0) {
    printf("I am the child.\n");
} else {
    printf("I am the parent.\n");
}
```

- **exec()**

  Purpose: Replaces the current process image with a new process image.

  Used to run a new program in the current process space.

  Types (family): execl(), execp(), execv(), execve(), etc.

  Syntax (C): execl("/bin/ls", "ls", "-l", NULL);

  Key Point:

  - If successful, exec() does not return.
  - If it fails, returns -1.

- **wait()**

  Purpose: Makes a parent process wait until one of its child processes finishes.

  Syntax: pid_t pid = wait(int *status);

  Returns:

  - Process ID of the terminated child
  - -1 on error

  Example:

  C

  ```
  int status;

  wait(&status);
  ```

- **exit()**

  Purpose: Terminates the calling process and returns a status to the parent process.

  Syntax: exit(status_code);

  - status_code is usually 0 for success, or a custom integer.

  Example:

  C

  ```
  exit(0);
  ```

2. File Management System calls:-

- **open()**

  Purpose: Opens a file and returns a **file descriptor (int)**.

  Header: <fcntl.h>, <sys/types.h>, <sys/stat.h>, <unistd.h>

  Syntax (cpp): int open(const char *pathname, int flags, mode_t mode);

- **read()**

  Purpose: Opens a file and returns a **file descriptor (int)**.

  Syntax (cpp): ssize_t read(int fd, void *buf, size_t count);

- **write()**

  Purpose: Writes data to a file descriptor.

  Syntax (cpp): ssize_t write(int fd, const void *buf, size_t count);

- **close()**

  Purpose: Closes an open file descriptor.

  Syntax (cpp): int close(int fd);

C++ Example: File Handling using System Calls

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

int main() {
    const char* filename = "sample.txt";

    // 1. Open file for writing (create if it doesn't exist)
    int fd = open(filename, O_WRONLY | O_CREAT, 0644);
    if (fd < 0) {
        std::cerr << "Error opening file for writing.\n";
        return 1; }
```

```
    // 2. Write to file
    const char* message = "Hello, Linux system calls!";
    ssize_t bytesWritten = write(fd, message, strlen(message));
    std::cout << "Written " << bytesWritten << " bytes.\n";

    // 3. Close the file
    close(fd);

    // 4. Open file for reading
    fd = open(filename, O_RDONLY);
    if (fd < 0) {
        std::cerr << "Error opening file for reading.\n";
        return 1;
    }

    // 5. Read from file
    char buffer[100];
    ssize_t bytesRead = read(fd, buffer, sizeof(buffer) - 1);
    buffer[bytesRead] = '\0'; // Null-terminate the string

    std::cout << "Read content: " << buffer << std::endl;

    // 6. Close the file
    close(fd);

    return 0;
}
```

3. Device Management System calls:-

- **read() – Read from a Device**

Header File: <unistd.h>
Syntax: ssize_t read(int fd, void *buf, size_t count);
Purpose: Reads count bytes from file descriptor fd into the buffer buf.

Example:

CPP

```
#include <iostream>

#include <fcntl.h>

#include <unistd.h>

including namespace std;


int main() {
   int fd = open("/dev/tty", O_RDONLY);
   if (fd < 0) {
      cerr << "Failed to open /dev/tty\n";
      return 1;
   }


   char buffer[100];
   int bytes = read(fd, buffer, sizeof(buffer));
   buffer[bytes] = '\0';


   cout << "You typed: " << buffer << endl;
```

```
    close(fd);
    return 0;
}
```

- **write() – Write to a Device**

  Header File: <unistd.h>
  Syntax: ssize_t write(int fd, const void *buf, size_t count);
  Purpose: Writes count bytes from buf to file descriptor fd.

  Example:

  CPP

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

int main() {
    int fd = open("/dev/tty", O_WRONLY);
    if (fd < 0) {
        std::cerr << "Failed to open /dev/tty\n";
        return 1;
    }

    const char* message = "Hello from device writer!\n";
    write(fd, message, strlen(message));

    close(fd);
    return 0; }
```

- ## ioctl() – Device Control Operations

Header File: <sys/ioctl.h>
Syntax: int ioctl(int fd, unsigned long request, ...);
Purpose: Used for device-specific input/output operations.

Example:

CPP

```
#include <iostream>
#include <unistd.h>
#include <sys/ioctl.h>

int main() {
    if (isatty(STDIN_FILENO)) {
        std::cout << "stdin is a terminal device.\n";
    } else {
        std::cout << "stdin is not a terminal device.\n";
    }

    return 0;
}
```

- ## select() – Monitor Multiple Devices/File Descriptors

Header File: <sys/select.h>
Syntax: int select(int nfds, fd_set *readfds, fd_set *writefds,
fd_set *exceptfds, struct timeval *timeout);
Purpose: Waits for multiple file descriptors (like devices) to be ready.

Example:

CPP

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

int main() {
    fd_set readfds;
    FD_ZERO(&readfds);
    FD_SET(STDIN_FILENO, &readfds);

    struct timeval timeout = {10, 0};  // 10 seconds timeout

    std::cout << "Type something within 10 seconds: ";

    int result = select(STDIN_FILENO + 1, &readfds, NULL,
NULL, &timeout);

    if (result == -1) {
        std::cerr << "select() error\n";
    } else if (result == 0) {
        std::cout << "\nTimeout. No input detected.\n";
    } else {
        std::cout << "\nInput detected!\n";
    }

    return 0;
}
```

4. System Information Management System calls :-

- **socket() – Create a Socket**
  Header File: <sys/socket.h>

  Purpose: Creates an endpoint for communication and returns a socket descriptor.

  Syntax: int socket(int domain, int type, int protocol);

  Example:

  CPP

```
#include <iostream>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main() {
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        std::cerr << "Socket creation failed.\n";
        return 1;
    }
    std::cout << "Socket created successfully.\n";
    return 0;
}
```

- **connect() – Connect to a Remote Server**
  Header File: <sys/socket.h>

  Purpose: Connects the socket to the specified server address.

  Syntax: int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);

Example:

CPP

```
#include <iostream>
#include <cstring>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>

int main() {
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        std::cerr << "Socket creation failed.\n";
        return 1;
    }

    sockaddr_in server_addr;
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(80); // HTTP port
    inet_pton(AF_INET, "93.184.216.34", &server_addr.sin_addr);
// example.com

    if (connect(sockfd, (sockaddr*)&server_addr,
sizeof(server_addr)) < 0) {
        std::cerr << "Connection failed.\n";
        return 1;
    }
```

```
    std::cout << "Connected to server!\n";
    close(sockfd);
    return 0;
}
```

- **send() – Send Data to Server**

  Header File: <sys/socket.h>

  Purpose: Sends data on the connected socket.

  Syntax: ssize_t send(int sockfd, const void *buf, size_t len, int flags);

  Example:

  CPP

```
#include <iostream>
#include <cstring>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>

int main() {
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    sockaddr_in server_addr;
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(80);
    inet_pton(AF_INET, "93.184.216.34", &server_addr.sin_addr);
// example.com
    connect(sockfd, (sockaddr*)&server_addr, sizeof(server_addr));

    const char* httpRequest = "GET / HTTP/1.1\r\nHost:
example.com\r\n\r\n";
```

```
    send(sockfd, httpRequest, strlen(httpRequest), 0);


    std::cout << "HTTP GET request sent.\n";
    close(sockfd);
    return 0;
}
```

- **recv() – Receive Data from Server**

  Header File: <sys/socket.h>

  Purpose: Receives data from a connected socket.

  Syntax: ssize_t recv(int sockfd, void *buf, size_t len, int flags);

  Example:

  CPP

```
#include <iostream>
#include <cstring>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>


int main() {
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    sockaddr_in server_addr;
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(80);
    inet_pton(AF_INET, "93.184.216.34", &server_addr.sin_addr);
    connect(sockfd, (sockaddr*)&server_addr, sizeof(server_addr));
    const char* httpRequest = "GET / HTTP/1.1\r\nHost:
example.com\r\n\r\n";
```

```
    send(sockfd, httpRequest, strlen(httpRequest), 0);

    char buffer[4096];
    int bytesReceived = recv(sockfd, buffer, sizeof(buffer) - 1, 0);
    buffer[bytesReceived] = '\0';

    std::cout << "Received:\n" << buffer << std::endl;

    close(sockfd);
    return 0;
}
```

5. Network Management System calls :-

- **getpid() – Get Process ID**

Header File: <unistd.h>

Purpose: Returns the **Process ID** of the calling process.

Syntax: pid_t getpid();

Example:

CPP

```
#include <iostream>
#include <unistd.h>
int main() {
    pid_t pid = getpid();
    std::cout << "Current Process ID: " << pid << std::endl;
    return 0;
}
```

- **getuid() – Get User ID**

  Header File: <unistd.h>

  Purpose: Returns the **real user ID** of the calling process

  Syntax: uid_t getuid();

  Example:

  CPP

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

int main() {
    uid_t uid = getuid();
    std::cout << "User ID: " << uid << std::endl;
    return 0;
}
```

- **gethostname() – Get System Hostname**

  Header File: <unistd.h>

  Purpose: Retrieves the **standard host name** for the current machine.

  Syntax: int gethostname(char *name, size_t len);

  Example:

  CPP

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

int main() {
    char hostname[1024];
```

```
    gethostname(hostname, sizeof(hostname));
    std::cout << "Hostname: " << hostname << std::endl;
    return 0;
}
```

- **sysinfo() – Get System Information**
- **Header File:** <sys/sysinfo.h>

Purpose: Returns various system statistics like uptime, load, memory usage, etc.

Syntax: int sysinfo(struct sysinfo *info);

Example:

CPP

```
#include <iostream>
#include <sys/sysinfo.h>

int main() {
    struct sysinfo info;
    if (sysinfo(&info) == 0) {
        std::cout << "System Uptime (seconds): " << info.uptime << std::endl;
        std::cout << "Total RAM: " << info.totalram / (1024 * 1024) << " MB" << std::endl;
        std::cout << "Free RAM: " << info.freeram / (1024 * 1024) << " MB" << std::endl;
    } else {
        std::cerr << "sysinfo call failed!" << std::endl;
    }
    return 0;}
```