System Call

1. Name: fork()

Description: The fork() system call creates a new process by duplicating the calling process.

Header file: <unistd.h>

Syntax: pid_t fork(void);

Arguments: None.

Return type:

- On success: In the parent process, the process ID of the child process is returned. In the child process, 0 is returned.
- On failure: -1 is returned and errno is set appropriately.

2. Name: execl()

Description: The execl() system call replaces the current process image with a new process image specified by the path name.

Header file: <unistd.h>

Syntax: int execl(const char *path, const char *arg, ...);

Arguments:

- *path: Pointer to a null-terminated string that specifies the path name of the new process image file.
- *arg: Pointer to a null-terminated string that specifies the name of the new process image.
- ...: A list of zero or more additional arguments to be passed to the new process.

Return type:

- On success: This function does not return, as the calling process is replaced by the new process image.
- On failure: -1 is returned and errno is set appropriately.

3. Name: getpid()

Description: The getpid() system call returns the process ID of the calling process.

Header file: <unistd.h>

Syntax: pid_t getpid(void);

Arguments: None.

Return type:

- On success: The process ID of the calling process is returned.
- On failure: -1 is returned and errno is set appropriately.

4. Name: getppid()

Description: The getppid() system call returns the parent process ID of the calling process.

Header file: <unistd.h>

Syntax: pid_t getppid(void);

Arguments: No arguments.

Return type:

- On success: The parent process ID of the calling process is returned.
- On failure: -1 is returned and errno is set appropriately.

5. Name: exit()

Description: The exit() system call terminates the calling process immediately, and returns an exit status to the parent process.

Header file: <stdlib.h>

Syntax: void exit(int status);

Arguments: status: An integer value that specifies the exit status of the process.

Return type:

- This function does not return.

6. Name: wait()

Description: The wait() system call suspends the execution of the calling process until one of its child processes terminates.

Header file: <sys/wait.h>

Syntax: pid_t wait(int *status);

Arguments:*status: Pointer to an integer in which the exit status of the terminated child process is stored.

Return type:

- **On success:** The process ID of the terminated child.
- **On failure:** -1 is returned if there are no child processes, if the calling process has no children, or if the call is interrupted by a signal.

7. Name: close()

Description: The close() system call closes a file descriptor, so that it no longer refers to any file and can be reused.

Header file: <unistd.h>

Syntax: int close(int fd);

Arguments: fd: The file descriptor to be closed.

Return type:

- On success: 0 is returned.
- On failure: -1 is returned if the file descriptor is not valid or the operation is not permitted.

8. Name: opendir()

Description: The opendir() system call opens a directory stream corresponding to the directory given by the path name.

Header file: <dirent.h>

Syntax: DIR *opendir(const char *name);

Arguments: *name: Pointer to a null-terminated string that specifies the name of the directory.

Return type:

- On success: A pointer to a DIR structure representing the directory stream is returned.
- On failure: NULL is returned if the directory does not exist or if permission is denied.

9. Name: readdir()

Description: The readdir() system call reads the next directory entry from the directory stream referred to by the DIR pointer.

Header file: <dirent.h>

Syntax: struct dirent *readdir(DIR *dirp);

Arguments:*dirp: Pointer to a DIR structure representing the directory stream.

Return type:

- On success: A pointer to a dirent structure containing the directory entry information is returned.
- On failure: NULL is returned at the end of the directory stream or if an error occurs.

10. Name: open()

Description: The open() system call opens a file or device and returns a file descriptor.

Header file: <fcntl.h>

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

Arguments: *path: Pointer to a null-terminated string that specifies the file or device to be opened.

- flags: A bitwise OR of the access modes and file status flags.
- mode: A value specifying the permissions to be set for the file, in case it is created.

Return type:

- On success: A file descriptor is returned.
- On failure: -1 is returned and errno is set appropriately.

11. Name: read()

Description: The read() system call reads data from a file descriptor into a buffer.

Header file: <unistd.h>

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

Arguments:

- fd: The file descriptor from which to read data.

- *buf: Pointer to the buffer where the data read will be stored.
- count: The maximum number of bytes to read.

Return type:

- On success: The number of bytes read is returned.
- On failure: -1 is returned and errno is set appropriately.

12. Name: write()

Description: The write() system call writes data to a file descriptor from a buffer.

Header file: <unistd.h>

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

Arguments:

- fd: The file descriptor to which to write data.
- *buf: Pointer to the buffer containing the data to be written.
- count: The number of bytes to be written.

Return type:

- On success: The number of bytes written is returned.
- On failure: -1 is returned and errno is set appropriately.

13. Name: creat()

Description: The creat() system call creates a new file or opens an existing file with write-only access.

Header file: <fcntl.h>

Syntax: int creat(const char *path, mode_t mode);

Arguments:

- *path: Pointer to a null-terminated string that specifies the name of the file to be created.
- mode: A value specifying the permissions to be set for the file.

Return type:

- On success: A file descriptor is returned.
- On failure: -1 is returned and errno is set appropriately.

14 Name: sleep()

Description: The sleep() system call suspends execution of the calling process for a specified number of seconds.

Header file: <unistd.h>

Syntax: unsigned int sleep(unsigned int seconds);

Arguments: seconds: The number of seconds to suspend execution.

Return type:

- On success: The number of seconds remaining to sleep is returned (0 if the sleep completed).
- On failure: The sleep function always completes successfully and returns a value of 0.