System commands

1. Name: cp

Purpose: The cp command is used to copy files or directories.

Options:

-r: copies directories recursively

-i: prompts before overwriting files

-u: copies only when the source file is newer than the destination file

-p: preserves the file attributes, such as permissions and timestamps

-v: displays verbose output, showing each file as it is copied

Syntax: cp [OPTIONS] source_file destination_file

Example: cp file1.txt file2.txt

2. Name: mv

Purpose: The mv command is used to move or rename files or directories.

Options:

-i: prompts before overwriting files

-u: moves only when the source file is newer than the destination file

-v: displays verbose output, showing each file as it is moved

Syntax: mv [OPTIONS] source_file destination_file

Example: mv file1.txt file2.txt

3. Name: ls

Purpose: The ls command is used to list the contents of a directory.

Options:

-l: displays the long format, showing detailed file information

-a: shows hidden files and directories

-h: shows file sizes in a human-readable format

-t: sorts the files by modification time

-r: reverses the order of the sort

Syntax: ls [OPTIONS] [FILE]...

Example: ls -l

4. Name: grep

Purpose: The grep command is used to search for a pattern in a file or input.

Options:

-i: ignores case when searching

-v: displays the lines that do not match the pattern

-c: displays the count of matches -n: displays the line numbers of matches

-r: searches recursively in directories

Syntax: grep [OPTIONS] pattern [FILE]...

Example: grep "hello" file1.txt

5. Name: chmod

Purpose: The chmod command is used to change the permissions of a file or directory.

Options:

u/g/o/a: specifies the user, group, others, or all

+/-/=: adds, removes, or sets the permissions

r/w/x: specifies read, write, or execute permissions

Syntax: chmod [OPTIONS] mode file/directory

Example: chmod u+x file1.txt

6. Name: cat

Purpose: The cat command is used to display the contents of a file.

Options:

-n: displays line numbers

-b: displays line numbers for non-blank lines

-s: squeezes consecutive blank lines into one

Syntax: cat [OPTIONS] [FILE]...

Example: cat file1.txt

7. Name: mkdir

Purpose: The mkdir command is used to create a new directory.

Options:

-p: creates parent directories if they do not exist

-v: displays verbose output, showing each directory as it is created

Syntax: mkdir [OPTIONS] directory

Example: mkdir new_directory

8. Name: rm

Purpose: The rm command is used to remove files or directories.

Options:

-i: prompts before deleting each file

-r: deletes directories and their contents recursively

-f: forces the deletion without prompting

Syntax: rm [OPTIONS] file/directory

Example: rm file1.txt

9. Name: rmdir

Purpose: The rmdir command is used to remove empty directories.

Options:

-p: removes parent directories if they become empty

-v: displays verbose output, showing each directory as it is removed

Syntax: rmdir [OPTIONS] directory

Example: rmdir empty_directory

10. Name: wc

Purpose: The wc command is used to count the number of words, lines, and characters in a file or input.

Options:

-l: displays the count of lines

-w: displays the count of words

-c: displays the count of bytes

Syntax: wc [OPTIONS] [FILE]...

Example: wc -l file1.txt

11. Name: who

Purpose: The who command is used to display information about currently logged in users.

Options:

-a: displays information about idle time and the process being run

-r: displays the current runlevel

Syntax: who [OPTIONS]

Example: who -a

12. Name: head

Purpose: The head command is used to display the first few lines of a file.

Options:

-n: specifies the number of lines to display

Syntax: head [OPTIONS] [FILE]...

Example: head -n 5 file1.txt

13. Name: tail

Purpose: The tail command is used to display the last few lines of a file.

Options:

-n: specifies the number of lines to display

-f: displays the lines as they are appended to the file

Syntax: tail [OPTIONS] [FILE]...

Example: tail -n 5 file1.txt

14. Name: nl

Purpose: The nl command is used to display the contents of a file with line numbers.

Options:

-b: specifies when to number the lines

-b a: numbers all lines

-b t: numbers non-blank lines

-n: specifies the line numbering format

Syntax: nl [OPTIONS] [FILE]...

Example: nl -b a file1.txt

15. Name: awk

Purpose: The awk command is used to search for and manipulate text in a file.

Options:

-F: specifies the field separator

-v: defines a variable

-f: specifies a script file to execute

Syntax: awk [OPTIONS] 'pattern {action}' [FILE]...

Example: awk -F, '{print $1}' file1.txt

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.