1) Demonstrate Orphan process

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

pid\_t child\_pid = fork(); // Create a child process

if (child\_pid < 0) { // fork failed

perror("fork failed");

exit(EXIT\_FAILURE);

}

if (child\_pid == 0) { // Child process

printf("Child process with PID: %d\n", getpid());

sleep(2); // Child process sleeps for 2 seconds

printf("Child process exiting.\n");

} else { // Parent process

printf("Parent process with PID: %d\n", getpid());

printf("Parent process exiting.\n");

exit(EXIT\_SUCCESS); // Parent process exits

}

return 0;

}

2) Demonstrate zombie process

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

int main() {

pid\_t child\_pid = fork(); // Create a child process

if (child\_pid < 0) { // fork failed

perror("fork failed");

exit(EXIT\_FAILURE);

}

if (child\_pid == 0) { // Child process

printf("Child process with PID: %d\n", getpid());

printf("Child process exiting.\n");

exit(EXIT\_SUCCESS);

} else { // Parent process

printf("Parent process with PID: %d\n", getpid());

sleep(5); // Parent process waits for 5 seconds

printf("Parent process exiting.\n");

// Parent process doesn't wait for the child to exit

}

return 0;

}

3) Program to demonstrate the use of execl (execute followed by l which stands for list of arguments)

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main() {

printf("Executing ls command...\n");

// Using execl to replace the current process image with 'ls' command

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

// If execl returns, it means an error occurred

perror("execl failed");

return EXIT\_FAILURE;

}

4) Program to demonstrate the use of execlp (t is used to execute a file specified by its name)

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main() {

printf("Executing ls command using execlp...\n");

// Using execlp to replace the current process image with 'ls' command

execlp("ls", "ls", "-l", NULL);

// If execlp returns, it means an error occurred

perror("execlp failed");

return EXIT\_FAILURE;

}

5) c program to demonstrate use of execle (executable path and environment variables directly.)

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main() {

printf("Executing ls command using execle...\n");

// Setting environment variables

char \*envp[] = {"PATH=/bin", NULL};

// Using execle to replace the current process image with 'ls' command

execle("/bin/ls", "ls", "-l", NULL, envp);

// If execle returns, it means an error occurred

perror("execle failed");

return EXIT\_FAILURE;

}

6) Demonstrate use of execv (execute a file with a vector of arguments)

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main() {

printf("Executing ls command using execv...\n");

// Arguments for the ls command

char \*args[] = {"ls", "-l", NULL};

// Using execv to replace the current process image with 'ls' command

execv("/bin/ls", args);

// If execv returns, it means an error occurred

perror("execv failed");

return EXIT\_FAILURE;

}

7) Demonstrate the use of execvp (execute file with variable arguments and path)

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main() {

printf("Executing ls command using execvp...\n");

// Arguments for the ls command

char \*args[] = {"ls", "-l", NULL};

// Using execvp to replace the current process image with 'ls' command

execvp("ls", args);

// If execvp returns, it means an error occurred

perror("execvp failed");

return EXIT\_FAILURE;

}

8) Demonstrate execvpe(execute file with vector array of arguments, path environment variable and environmental variables)

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#define \_GNU\_SOURCE

int main() {

printf("Executing ls command using execvpe...\n");

// Arguments for the ls command

char \*args[] = {"ls", "-l", NULL};

// Setting environment variables

char \*envp[] = {"PATH=/bin", NULL};

// Using execvpe to replace the current process image with 'ls' command

execvpe("ls", args, envp);

// If execvpe returns, it means an error occurred

perror("execvpe failed");

return EXIT\_FAILURE;

}