***System Calls in Process Management***

1. **fork():** This system call is used to create a new process by duplicating the calling process. The new process, called the child process, is an exact copy of the calling process, called the parent process.

Code 🡪 #include <stdio.h>

#include <unistd.h>

int main() {

pid\_t pid;

pid = fork();

if (pid < 0) {

fprintf(stderr, "Fork failed\n");

return 1;

} else if (pid == 0) {

// Child process

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

} else {

// Parent process

printf("Parent process, Child PID: %d\n", pid);

}

return 0;

}

1. **exec():** This system call is used to replace the current process with a new process. It loads a new program into the current process space, replacing the current program with a new one.

Code 🡪 #include <stdio.h>

#include <unistd.h>

int main() {

printf("Before exec()\n");

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

printf("After exec(), this line should not be reached\n");

return 0;

}

1. **wait():** This system call is used to suspend execution of the calling process until one of its child processes terminates.

Code 🡪 #include <stdio.h>

#include <stdlib.h>

#include <sys/wait.h>

#include <unistd.h>

int main() {

pid\_t pid;

pid = fork();

if (pid < 0) {

fprintf(stderr, "Fork failed\n");

return 1;

} else if (pid == 0) {

// Child process

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

sleep(2); // Simulating some work in child process

} else {

// Parent process

printf("Parent process, Child PID: %d\n", pid);

wait(NULL); // Wait for child process to finish

printf("Child process has terminated\n");

}

return 0;

}

1. **exit():** This system call is used to terminate the calling process and return an exit status to the parent process.

Code 🡪 #include <stdio.h>

#include <stdlib.h>

int main() {

printf("Before exit()\n");

exit(0);

printf("After exit(), this line should not be reached\n");

return 0;

}

1. **getpid():** This system call is used to retrieve the process ID of the calling process.

Code 🡪 #include <stdio.h>

#include <unistd.h>

int main() {

printf("Process PID: %d\n", getpid());

return 0;

}

1. **kill():** This system call is used to send a signal to a process or a group of processes specified by their process ID.

Code 🡪 #include <stdio.h>

#include <signal.h>

#include <unistd.h>

int main() {

// Assuming a process with PID 1234 exists

pid\_t pid = 1234;

kill(pid, SIGKILL);

printf("Sent SIGKILL signal to process with PID %d\n", pid);

return 0;

}