#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>

#include <sys/types.h>

pid\_t child\_pid;

void sigchld\_handler(int sig) {

int status;

if (waitpid(child\_pid, &status, WNOHANG) > 0) {

if (WIFEXITED(status)) {

printf("Child process exited with status %d\n", WEXITSTATUS(status));

} else if (WIFSIGNALED(status)) {

printf("Child process terminated by signal %d\n", WTERMSIG(status));

}

}

}

void sigalrm\_handler(int sig) {

printf("Alarm signal received: 5 seconds timeout reached!\n");

if (child\_pid > 0) {

kill(child\_pid, SIGKILL);

printf("Child process killed due to timeout\n");

}

}

int main() {

struct sigaction sa\_sigchld, sa\_sigalrm;

sa\_sigchld.sa\_handler = sigchld\_handler;

sa\_sigchld.sa\_flags = SA\_RESTART;

sigemptyset(&sa\_sigchld.sa\_mask);

sigaction(SIGCHLD, &sa\_sigchld, NULL);

sa\_sigalrm.sa\_handler = sigalrm\_handler;

sa\_sigalrm.sa\_flags = SA\_RESTART;

sigemptyset(&sa\_sigalrm.sa\_mask);

sigaction(SIGALRM, &sa\_sigalrm, NULL);

child\_pid = fork();

if (child\_pid < 0) {

perror("Fork failed");

exit(1);

}

if (child\_pid == 0) {

printf("Child process: Executing a command (e.g., sleep 10)\n");

execlp("sleep", "sleep", "10", NULL);

} else {

printf("Parent process: Setting an alarm for 5 seconds...\n");

alarm(5);

pause();

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

}

return 0;

}