MANAS LAUD

22BKT0060

OPERATING SYSTEMS LAB CLASS DA-1

Q1. Write a C Program X that create two child process Y and Z and write a program A that creates child process

B such that the child B loads and executes the program X. You should make each of the process to do some

I/O Operations like Waiting for some input or writing something to the output screen

Code.

Filename: X.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

void childY() {

printf("Child Y: Waiting for input...\n");

char buffer[100];

fgets(buffer, sizeof(buffer), stdin);

printf("Child Y: You entered: %s\n", buffer);

}

void childZ() {

printf("Child Z: Writing output...\n");

printf("Child Z: This is a message from Child Z.\n");

}

int main() {

pid\_t pidY, pidZ;

pidY = fork();

if (pidY < 0) {

perror("Failed to fork Y");

exit(1);

} else if (pidY == 0) {

childY();

exit(0);

} else {

pidZ = fork();

if (pidZ < 0) {

perror("Failed to fork Z");

exit(1);

} else if (pidZ == 0) {

childZ();

exit(0);

} else {

wait(NULL);

wait(NULL);

}

}

printf("Parent of Y and Z: Both children have finished.\n");

return 0;

}

Filename: A.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

int main() {

pid\_t pidB;

pidB = fork();

if (pidB < 0) {

perror("Failed to fork B");

exit(1);

} else if (pidB == 0) {

execl("./X", "X", (char \*)NULL);

perror("execl failed");

exit(1);

} else {

wait(NULL);

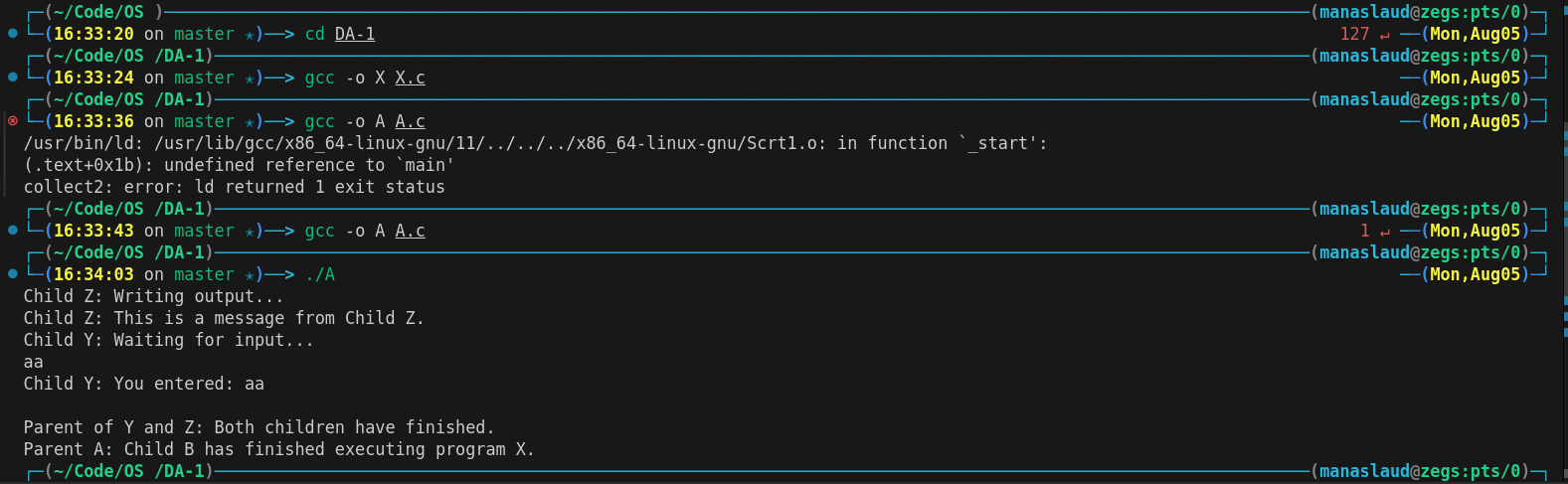
printf("Parent A: Child B has finished executing program X.\n");

}

return 0;

}

Output:

Q2.Write two programs : one called client.c, the other called server.c. The client program lists a

prompter and reads from the keyboard two integers and one of the characters '+' or '-'. The read

information is transmitted with the help of the system call execl to a child process, which executes

the server code. After the child (server) process finishes the operation, it transmits the result to

parent process (client) with the help of the system call exit. The client process prints the result on

the screen and also reprints the prompter, ready for a new reading.

Code.

Filename: server.c

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char \*argv[]) {

if (argc != 4) {

fprintf(stderr, "Usage: %s <int1> <int2> <operator>\n", argv[0]);

exit(1);

}

int num1 = atoi(argv[1]);

int num2 = atoi(argv[2]);

char operator = argv[3][0];

int result;

switch (operator) {

case '+':

result = num1 + num2;

break;

case '-':

result = num1 - num2;

break;

default:

fprintf(stderr, "Invalid operator. Use '+' or '-'.\n");

exit(1);

}

printf("Server: Calculated result = %d\n", result);

exit(result);

}

Filename: client.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

int main() {

while (1) {

int num1, num2;

char operator;

printf("Enter two integers and an operator (+ or -): ");

if (scanf("%d %d %c", &num1, &num2, &operator) != 3) {

fprintf(stderr, "Invalid input. Try again.\n");

while (getchar() != '\n');

continue;

}

pid\_t pid = fork();

if (pid < 0) {

perror("Failed to fork");

exit(1);

} else if (pid == 0) {

char num1\_str[20], num2\_str[20];

snprintf(num1\_str, sizeof(num1\_str), "%d", num1);

snprintf(num2\_str, sizeof(num2\_str), "%d", num2);

execl("./server", "server", num1\_str, num2\_str, &operator, (char \*)NULL);

perror("execl failed");

exit(1);

} else {

int status;

wait(&status);

if (WIFEXITED(status)) {

int result = WEXITSTATUS(status);

printf("Result: %d\n", result);

} else {

printf("Child process did not terminate normally.\n");

}

}

}

return 0;

}

Output:

