ОС ДЗ 3

Власов Николай Алексеевич, БПИ229

8 февраля 2024 г.

Описание кода

Сделал на 10 :)
Макрос $\#define\ num\ uint64\_t$ нужен чтобы быстрее писать код в 64-разрядной арифметике
Функции $num\ factorial(num\ n)$ и $num\ fibbonacci(num\ n)$ считают факториал числа n и n-ное число Фибоначчи соответственно.
В функции $main$ сначала проверяется наличие поданого в качестве аргумента командной строки числа N, затем проверяется, что подается неотрицательное число. После проверок создается с помощью $fork()$ процесс-ребенок, который будет вычислять значение факториала. Затем с помощью условных операторов мы понимаем, в каком процессе сейчас находимся и производим нужные вычисления. **(8 баллов)**
Хочется отметить, что в родительском процессе мы ждем окончания процесса-ребенка, и только после этого печатаем содержимое текущей директории **(+2 балла)**

Код программы

Листинг 1: Директивы препроцессора и factorial

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdint.h>

#define num uint64_t

// function for counting n!
num factorial(num n) {
    num result = 1;
    for (num i = 2; i <= n; ++i) {
        if (result * i < result) {
            fprintf(stderr, "Overflow of n! in n = %lu\n", n);
            return 0;
        }

        result *= i;
    }

    return result;
}
```

Листинг 2: main

```
1  int main(int argc, char *argv[]) {
2      if (argc == 1) {
   // there is no arguments except executable file name
3          fprintf(stderr, "Enter␣N␣as␣a␣command-line␣argument\n");
4          return -1;
5      }
6
7      if (argv[1][0] == '-') {
8          fprintf(stderr, "Enter␣positive␣N␣as␣a␣command-line␣argument\n");
9          return -1;
10     }
11
12     num n = atoi(argv[1]);
13
14     pid_t process = fork();
   // fork process
15
16     if (process < 0) {
   // error during process creating
17         fprintf(stderr, "Error␣during␣process␣creating\n");
18         return -1;
19     }
20
21     if (process == 0) {
   // process for n! counting
22         printf("Child␣process,␣PID:␣%d;\tParent␣process:␣%d\n", getpid(), getppid());
23         num fact = factorial(n);
24         if (fact == 0) {  // overflow
25             return -1;
26         }
27         printf("%lu!␣=␣%lu\n", n, fact);
28     } else {
   // process for F_n counting
29         printf("Parent␣process,␣PID:␣%d;\tChild␣process:␣%d\n", getpid(), process);
30         num fibbo = fibbonacci(n);
31         if (n == 0 || fibbo > 0) {  // overflow
32             printf("F_%lu␣=␣%lu\n", n, fibbo);
33         }
34         waitpid(process, NULL, 0);
   // wait while child process end
35         execlp("ls", "ls", "-l", (char *) NULL);
   // print inforamtion about current dir
36     }
37
38     return 0;
39 }
```

Листинг 3: main.c

```
1  // function for counting F_n
2  num fibbonacci(num n) {
3      if (n <= 1) {
4          return n;
5      }
6
7      num previous = 0, current = 1, tmp = -1;
8      for (unsigned int i = 2; i <= n; ++i) {
9          if (previous + current < current) {
10             fprintf(stderr, "Overflow of F_n in n = %lu\n", n);
11             return 0;
12         }
13         tmp = previous;
14         previous = current;
15         current = current + tmp;
16     }
17
18     return current;
19 }
```