```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

/*|Matheus Martins Batista (2019005687) - Sistemas Operacionais (COM120) |
  |Ciências da Computação (CCO) - EP02 - Exercício01 - 24/09/2021        |*/

void verifica_fork(int *pid){
    if(pid<0){
        perror("Fork: ");
        exit(1);
    }
}

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

    int pid = 0, i;
    int pidpai, pidCA, pidCB, pidCC, pidCD; /*Guardar pid do pai e dos filhos C(child)*/
    pidpai = getpid();
    pid = fork();
    verifica_fork(&pid); /*Conferir se o fork conseguiu criar um novo processo*/

/*Verificar se o processo rodando é o pai ou filho com base no pid(fork do filho retorna 0)*/

    if (pid == 0){
        pidCA = getpid();
        fork();
        /*Verificar qual filho está executando e garantir que CA crie apenas CC e CD*/
        if(getpid() == pidCA){
            fork();
            if(getpid() == pidCA){ //CA já criou os filhos, pode começar a contar
                for(i=1000;i<2000;i++){
                    printf("Filho CA(%d) contando: %d\n", pidCA, i);
                }
            }
            else{
                pidCD = getpid();
                verifica_fork(&pidCD);
                for(i=4000;i<5000;i++){
                    printf("Filho CD(%d) contando: %d\n", pidCD,i);
                }

            }
        }
        /* Se CA não está em execução, CC fica livre para iniciar a contagem*/
        else{
            pidCC = getpid();
            verifica_fork(&pidCC);
            for(i=3000;i<4000;i++){
                printf("Filho CC(%d) contando: %d\n", pidCC, i);
            }
        }

    }
    /*Execução do pai e o filho CB é criado*/
    else{
        pid = fork();
        verifica_fork(&pid);
        if(pid==0){
            pidCB = getpid();
```

```c
            for(i=2000;i<3000;i++){
                printf("Filho CB(%d) contando: %d\n", pidCB, i);
            }
        }
        else{
            for(i=0;i<1000;i++){
                printf("Pai(%d) contando: %d\n", pidpai, i);
            }
        }
    }

    return 0;
}
/*As condicionais garantem a criação dos filhos antes do processo de contagem, contudo sem uma função
que controle a ordem de execução dos processos as contagens são feitas ao mesmo tempo, intercalando as
impressões no terminal de forma desordenada.*/
```

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

/*|Matheus Martins Batista (2019005687) - Sistemas Operacionais (COM120) |
  |Ciências da Computação (CCO) - EP02 - Exercício02 - 25/09/2021        |*/

double calcula_senx(double x, double xp ){ //Função para calcular o valor do seno
    double senx, ant, prox, prec;
    int i = 1;
    senx = x;
    do{
        ant = senx;
        senx = senx*(1-(xp/pow(i, 2)));
        prox = senx;
        //Pegar o valor absoluto para comparar a precisão
        if(ant>prox){
            prec = ant - prox;
        }
        else{
            prec = prox - ant;
        }
        i++;
    }while(prec>=pow(10,-12));

    return senx;
}

double calcula_cosx(double xp){ //Função para calcular o valor do cosseno
    double cosx = 1, ant, prox, prec;
    int i = 0;
    do{
        ant = cosx;
        cosx = cosx*(1-(4*xp/pow((2*i+1), 2)));
        prox = cosx;
        //Pegar o valor absoluto para comparar a precisão
        if(ant>prox){
            prec = ant - prox;
        }
        else{
            prec = prox - ant;
        }
        i++;
    }while(prec>=pow(10,-12)); //Se o valor absoluto for menor que precisão ele para

    return cosx;
}

void verifica_fork(int *pid){
    if(pid<0){
        perror("Fork: ");
        exit(1);
    }
}

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

    double x;
```

```c
    int pid = 0, pidC1 = 0, pidC2 = 0, cstatus;
    int fdc1[2], fdc2[2];                           //File descriptor para definir a saída e entrada nos pipes
    printf("Digite um ângulo em radiano entre 0 e π/2\n");
    scanf("%lf", &x);
    //Verificar se entrada é válida
    if(x>M_PI/2 || x<0){
        printf("Ângulo inválido! Fechando programa...\n");
        exit(1);
    }
    double xp = (x*x)/(M_PI*M_PI);                   //XP é um constante nos somatórios
    if (pipe(fdc1)<0 || pipe(fdc2)<0){              //Verificar erro nos pipes
        perror("Pipe: ");
        exit(1);
    }

    //Início da execução com processos pai e filho
    clock_t tic2 = clock(); //começa contagem

    pid = fork();
    verifica_fork(&pid);
    if(pid == 0){ //Filho escreve, fecha leitura
        double resultado_sen = 0;
        close(fdc1[0]);
        pidC1 = getpid();
        resultado_sen = calcula_senx(x, xp);
        write(fdc1[1], &resultado_sen, sizeof(resultado_sen));
        exit(0);
    }
    else{
        pid = fork();
        verifica_fork(&pid);
        if(pid == 0){ //Filho escreve, fecha leitura
            double resultado_cos = 0;
            close(fdc2[0]);
            pidC2 = getpid();
            resultado_cos = calcula_cosx(xp);
            write(fdc2[1], &resultado_cos, sizeof(resultado_cos));
            exit(0);
        }
        else{ //Pai calcula tangente
            double senx = 0, cosx = 0, tanx = 0;
            //Aguardar cálculo dos filhos
            waitpid(pidC1, &cstatus, 0);
            waitpid(pidC2, &cstatus, 0);
            //0 leitura, 1 escrita
            //Pai apenas lê, deve-se fechar a escrita
            close(fdc1[1]);
            close(fdc2[1]);
            read(fdc1[0], &senx, sizeof(senx));
            read(fdc2[0], &cosx, sizeof(cosx));
            tanx = senx/cosx;
            printf("A tangente é %.20lf\n", tanx);
            clock_t toc2 = clock(); //termina contagem
            double tempo_execucao2 = (double)(toc2-tic2) / CLOCKS_PER_SEC;
            printf("%fs decorridos com execução com processos\n", tempo_execucao2);


        }

    }
```

```c
    //Parte dedicada aos testes de precisão com tan() da math.h e análise do tempo com execução linear
    /*printf("A tan(x) da math.h calcula o valor: %.20lf \n", tan(x));

    double senx = 0, cosx = 0, tanx = 0;
    clock_t tic = clock();
    senx = calcula_senx(x, xp);
    cosx = calcula_cosx(xp);
    tanx = senx/cosx;
    printf("A tangente é %.20lf\n", tanx);
    clock_t toc = clock();
    double tempo_execucao = (double)(toc-tic) / CLOCKS_PER_SEC;
    printf("%fs decorridos com execução linear\n", tempo_execucao);*/

    return 0;
}

/*O programa foi escrito com funções para calcular o seno e cosseno com base nas fórmulas
fornecidas. A utilização de pipes foi essencial para garantir a comunicação entre pai (lê as
informações) e filhos(escreve o resultado do cálculo), surgiu a dúvida se há a necessidade da
utilização da waitpid() para garantir que o pai espere os filhos terminarem a execução, uma
vez que o programa também funcionou sem a utilização dessa função. Utilizando a biblioteca
math.h foi possível perceber que a precisão da fórmula somatória é muito próxima ao ressultado
obtido pela função tan(). Logo, o programa é capaz de calcular as funções seno, cosseno e
tangente com precisão. Ademais, a função clock() foi responsável por mensurar o tempo de
execução, rodando linearmente, o programa demora mais para executar o cálculo e é possível fazer
o teste, só é necessário retirar o comentário da parte dedicada aos testes.*/
```