Relatório Trabalho 3 – Sistemas Operacionais

Aluno: Otávio Malta Borges

Matricula: 12011BSI291

EX6) Escrever um programa multithreads que calcule vários valores estáticos para uma lista de números passados via linha de comando. Calcular a média, mínimo e máximo.

```c
static int *val = NULL;
static int tam = 0;

static int med;
static int min;
static int max;

int valores(int argc, char **argv);
void *media(void *p);
void *minimo(void *p);
void *maximo(void *p);

int main(int argc, char **argv){
    pthread_attr_t attr;
    pthread_t idMedia, idMin, idMax;
    int aux;
```

Primeiramente é criado as variáveis estáticas/globais med(média), min(mínimo), max(máximo), val(vetor cujo receberá os valores), tam(quantidade de argumentos).

A função main recebe como parâmetro argc e argv para receber os números via linha de comando.

É criado então, os atributos(attr) e os identificadores (idMedia, idMax e idMin) os quais serão usados na criação das threads. Além disso, é criado uma variável do tipo int, chamada aux para receber o retorno das funções e fazer a verificação de erro.

Em seguida é analisado se argc, o qual representa a quantidade de argumentos passados em argv, é igual a 1, para verificar se o usuário passou a lista de valores cujo será analisada.

Logo, é chamado a função valores, responsável por alocar um vetor (val) na memória a partir do malloc, transformar os valores de argv de String para inteiros(atoi) e por fim colocá-los em val. A função valores possui verificações caso argc seja 0 ou argv seja nulo ou caso a alocação de val tenha dado erro.

```c
int valores(int argc, char **argv){
    int i;
    tam = argc;

    if (argc <= 0 || argv == NULL){
        return -1;
    }

    val = (int*)malloc(argc * sizeof(*val));

    if (val == NULL){
        printf("Erro ao alocar os valores\n");
        return -1;
    }

    for (i = 0; i < argc; i++){
        val[i] = atoi(argv[i]);
    }
    return 0;
}
```

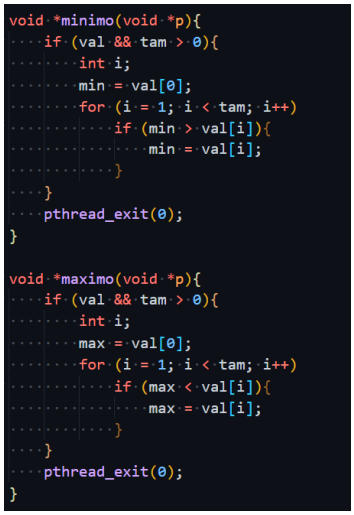
```c
if (pthread_attr_init(&attr) != 0) {
    printf("Erro ao iniciar atributos\n");
    return -1;
}

// MEDIA
aux = pthread_create(&idMedia, &attr, media, NULL);
if (aux != 0) {
    printf("Erro ao criar a thread da média!\n");
    return -1;
}
```

Então é inicializado os atributos que serão passados para as threads(pthread_attr_init) e em seguida as threads são criadas, passando os ids (criados inicialmente), os atributos, as respectivas funções (media, minimo, maximo) e os argumentos das funções, que nesse caso, são nulos.

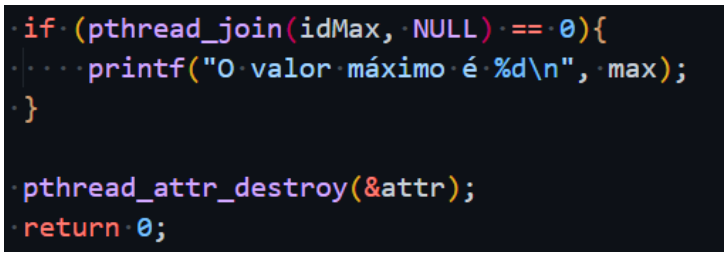
As funções mínimo e máximo, utilizadas nas threads, comparam cada valor no vetor global val com o valor mínimo/máximo atual. Caso o valor seja menor/maior que a variável global(min/max), então este é setado para a variável.

A função media, soma todos os valores do vetor val, e divide pela variável tam, cujo representa o tamanho do vetor. O resultado é colocado na variável global med.

As 3 funções verificam inicialmente caso o vetor e o tamanho sejam maiores que 0. E no final, chama a função pthread_exit para finalizar a thread e liberar os recursos utilizados.



```c
void *minimo(void *p){
    if (val && tam > 0){
        int i;
        min = val[0];
        for (i = 1; i < tam; i++)
            if (min > val[i]){
                min = val[i];
            }
    }
    pthread_exit(0);
}

void *maximo(void *p){
    if (val && tam > 0){
        int i;
        max = val[0];
        for (i = 1; i < tam; i++)
            if (max < val[i]){
                max = val[i];
            }
    }
    pthread_exit(0);
}
```



```c
if (pthread_join(idMax, NULL) == 0){
    printf("O valor máximo é %d\n", max);
}

pthread_attr_destroy(&attr);
return 0;
```

Por fim, é chamado a função pthread_join para cada thread criada, para que se espere a thread finalizar (passando o id como parâmetro), e então os atributos das threds é destruído com pthread_attr_destroy.

Compilando e executando:

```
otavio@otavio:/mnt/c/users/otavi/onedrive/ufu/4 º Período/so/trabalho3$ gcc -o ex6 ex6.c -pthread
otavio@otavio:/mnt/c/users/otavi/onedrive/ufu/4 º Período/so/trabalho3$ ./ex6
./ex6 [lista de valores]
otavio@otavio:/mnt/c/users/otavi/onedrive/ufu/4 º Período/so/trabalho3$ ./ex6 90 81 78 95 79 72 85
O valor médio é 82
O valor mínimo é 72
O valor máximo é 95
```

Tempo de execução:

```
otavio@otavio:/mnt/c/users/otavi/onedrive/ufu/4 º Período/so/trabalho3$ time ./ex6 90 81 78 95 79 72 85
O valor médio é 82
O valor mínimo é 72
O valor máximo é 95

real    0m0.109s
user    0m0.051s
sys     0m0.052s
otavio@otavio:/mnt/c/users/otavi/onedrive/ufu/4 º Período/so/trabalho3$ time ./ex6 90 81 78 95 79 72 85
O valor médio é 82
O valor mínimo é 72
O valor máximo é 95

real    0m0.129s
user    0m0.000s
sys     0m0.120s
otavio@otavio:/mnt/c/users/otavi/onedrive/ufu/4 º Período/so/trabalho3$ time ./ex6 90 81 78 95 79 72 85
O valor médio é 82
O valor mínimo é 72
O valor máximo é 95

real    0m0.014s
user    0m0.006s
sys     0m0.001s
```

EX7) Escrever um programa com thread que produza números primos até o valor passado pelo usuário via linha de comando.

```c
static int pri;

int main(int argc, char **argv){
    pthread_attr_t attr;
    pthread_t id;
    int aux;
```

Primeiro, é criado uma variável global pri, cujo receberá o calor digitado pelo usuário. A função main recebe como parâmetro argc e argv para receber os números via linha de comando.

É criado então, o atributo(attr) e o identificador (id) o qual será utilizado na criação da thread. Além disso, é criado uma variável do tipo int, chamada aux para receber o retorno das funções e fazer a verificação de erro.

Em seguida é analisado se argc, o qual representa a quantidade de argumentos passados em argv, é igual a 1, para verificar se o usuário passou o valor o qual será utilizado.

Logo é transformado o valor passado em argv para a variável global pri, utilizando a função atoi() para transformar a String em inteiro.

```c
    if (argc == 1){
            printf("%s [Valor]\n", *argv);
            return 1;
        }

    pri = atoi(argv[1]);
    if (pri < 0) {
        return 2;
    }
```

```c
if (pthread_attr_init(&attr) != 0) {
    printf("Erro ao iniciar atributos\n");
    return -1;
}

aux = pthread_create(&id, &attr, primos, NULL);
if (aux != 0) {
    printf("Erro ao criar a thread!\n");
    return -1;
}
```

Então é inicializado o atributo que será passado para a thread(pthread_attr_init) e em seguida a thread é criada, passando o id (criados inicialmente), o atributo, a função(primos) e o argumento da função, que nesse caso, é nulo.

A função primos() cria um laço de 2 (ignorando o 1, por não ser primo) até o número digitado (pri). Assim, é chamado a função ehprimo() para verificar se o valor atual do laço é um número primo e então printá-lo na tela. E no final, chama a função pthread_exit para finalizar a thread.

```c
void *primos(void *p) {
    for (int i = 2; i <= pri; i++) {
        if (ehprimo(i)){
            printf("%d ", i);
        }
    }
    pthread_exit(0);
}

int ehprimo(int n){
    for (int i = 2; i < n; i++)
        if (n % i == 0){
            return 0;
        }
    return 1;
}
```

A função ehprimo(), recebe um valor n como parâmetro, um laço de 2 até menor que n é criado, e então é comparado o resto da divisão de n pelo valor atual do laço, caso o número não seja divisível por nenhum número, a função retorna 1, indicando que n é primo.

```c
pthread_join(id, NULL);

pthread_attr_destroy(&attr);
return 0;
```

Por fim, é chamado a função pthread_join para que o programa espere a thread finalizar (passando o id como parâmetro), e então o atributo da thread é destruído com pthread_attr_destroy.

Compilando e executando:

```
otavio@otavio:/mnt/c/users/otavi/onedrive/ufu/4 º Período/so/trabalho3$ gcc -o ex7 ex7.c -pthread
otavio@otavio:/mnt/c/users/otavi/onedrive/ufu/4 º Período/so/trabalho3$ ./ex7
./ex7 [Valor]
otavio@otavio:/mnt/c/users/otavi/onedrive/ufu/4 º Período/so/trabalho3$ ./ex7 7
2 3 5 7
```

Tempo de execução:

```
otavio@otavio:/mnt/c/users/otavi/onedrive/ufu/4 º Período/so/trabalho3$ time ./ex7 7
2 3 5 7

real    0m0.014s
user    0m0.001s
sys     0m0.006s
otavio@otavio:/mnt/c/users/otavi/onedrive/ufu/4 º Período/so/trabalho3$ time ./ex7 7
2 3 5 7

real    0m0.011s
user    0m0.005s
sys     0m0.001s
otavio@otavio:/mnt/c/users/otavi/onedrive/ufu/4 º Período/so/trabalho3$ time ./ex7 7
2 3 5 7

real    0m0.015s
user    0m0.006s
sys     0m0.001s
```