Sistemas Operativos

2018/2019

Main

```c
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include "person.h"

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

    Person andre = new_person("Andre", 20);

    printf("idade anterior andre %d\n",andre.age);
    person_change_age(&andre, 30);
    printf("idade modificada andre %d\n",andre.age);

    Person new_andre = clone_person(&andre);

    person_change_age(&new_andre, 40);
    printf("idade andre %d\n", andre.age);
    printf("idade new_andre %d\n", new_andre.age);

    destroy_person(&new_andre);
    destroy_person(&andre);

    return 0;
}
```

Header File

```c
typedef struct Person{
    char* name;
    int age;
} Person;

//API
Person new_person(char* name, int age);
Person clone_person(Person* p);
void destroy_person(Person* p);
int person_age(Person* p);
void person_change_age(Person* p, int age);
```

Múltiplos “*.c*”

```c
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include "person.h"

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

    Person andre = new_person("Andre", 20);

    printf("idade anterior andre %d\n",andre.age);
    person_change_age(&andre, 30);
    printf("idade modificada andre %d\n",andre.age);

    Person new_andre = clone_person(&andre);

    person_change_age(&new_andre, 40);
    printf("idade andre %d\n", andre.age);
    printf("idade new_andre %d\n", new_andre.age);

    destroy_person(&new_andre);
    destroy_person(&andre);

    return 0;
}
```

```c
#include <stdio.h>
#include "person.h"
#include <stdlib.h>
#include <string.h>

Person new_person(char* name, int age) {
    // para ilustrar malloc + memcpy
    size_t n = strlen(name) + 1;
    char* s = malloc(sizeof(char[n]));
    memcpy(s, name, n);
    // ou simplesmente
    // char* s = strdup(name);
    return (Person) {
        .name = s,
        .age = age,
    };
}

Person clone_person(Person* p) {
    return (Person) {
        .name = strdup(p->name),
        .age = p->age,
    };
}
```

Makefile

```
CC = gcc
CFLAGS = -Wall

program: person
    $(CC) $(CFLAGS) program.c -o program person.o

person:
    $(CC) $(CFLAGS) -c person.c

clean:
    rm program person.o


```

Structs

```c
typedef struct Person{
    char* name;
    int age;
} Person;

//API
Person new_person(char* name, int age);
Person clone_person(Person* p);
void destroy_person(Person* p);
int person_age(Person* p);
void person_change_age(Person* p, int age);
```

Apontadores Vs Endereços

```c
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include "person.h"

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

    Person andre = new_person("Andre", 20);

    printf("idade anterior andre %d\n",andre.age);
    person_change_age(&andre, 30);
    printf("idade modificada andre %d\n",andre.age);

    Person new_andre = clone_person(&andre);

    person_change_age(&new_andre, 40);
    printf("idade andre %d\n", andre.age);
    printf("idade new_andre %d\n", new_andre.age);

    destroy_person(&new_andre);
    destroy_person(&andre);

    return 0;
}
```

```c
#include <stdio.h>
#include "person.h"
#include <stdlib.h>
#include <string.h>

Person new_person(char* name, int age) {
    // para ilustrar malloc + memcpy
    size_t n = strlen(name) + 1;
    char* s = malloc(sizeof(char[n]));
    memcpy(s, name, n);
    // ou simplesmente
    // char* s = strdup(name);
    return (Person) {
        .name = s,
        .age = age,
    };
}

Person clone_person(Person* p) {
    return (Person) {
        .name = strdup(p->name),
        .age = p->age,
    };
}
```

Scope

```c
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include "person.h"

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

    Person andre = new_person("Andre", 20);

    printf("idade anterior andre %d\n",andre.age);
    person_change_age(&andre, 30);
    printf("idade modificada andre %d\n",andre.age);

    Person new_andre = clone_person(&andre);

    person_change_age(&new_andre, 40);
    printf("idade andre %d\n", andre.age);
    printf("idade new_andre %d\n", new_andre.age);

    destroy_person(&new_andre);
    destroy_person(&andre);

    return 0;
}
```

```c
int person_age(Person* p){
    return p->age;
}

void person_change_age(Person* p, int age){
    p->age = age;
}
```

Memória Dinâmica

```c
Person new_person(char* name, int age) {
    // para ilustrar malloc + memcpy
    size_t n = strlen(name) + 1;
    char* s = malloc(sizeof(char[n]));
    memcpy(s, name, n);
    // ou simplesmente
    // char* s = strdup(name);
    return (Person) {
        .name = s,
        .age = age,
    };
}

Person clone_person(Person* p) {
    return (Person) {
        .name = strdup(p->name),
        .age = p->age,
    };
}

void destroy_person(Person* p) {
    free(p->name);
}
```

Outras Ferramentas

- Man
- gdb, lldb
- Valgrind