Sistemas Operativos

2018/2019

Main

```c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <strings.h>
4  #include "person.h"
5
6  int main(int argc, char* argv[]){
7
8      Person andre = new_person("Andre", 20);
9
10     printf("idade anterior andre %d\n",andre.age);
11     person_change_age(&andre, 30);
12     printf("idade modificada andre %d\n",andre.age);
13
14     Person new_andre = clone_person(&andre);
15
16     person_change_age(&new_andre, 40);
17     printf("idade andre %d\n", andre.age);
18     printf("idade new_andre %d\n", new_andre.age);
19
20     destroy_person(&new_andre);
21     destroy_person(&andre);
22
23     return 0;
24 }
```

Header File

```
1 typedef struct Person{
2     char* name;
3     int age;
4 } Person;
5
6 //API
7 Person new_person(char* name, int age);
8 Person clone_person(Person* p);
9 void destroy_person(Person* p);
10 int person_age(Person* p);
11 void person_change_age(Person* p, int age);
```

Múltiplos ".c"

```c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <strings.h>
4  #include "person.h"
5
6  int main(int argc, char* argv[]){
7
8      Person andre = new_person("Andre", 20);
9
10     printf("idade anterior andre %d\n",andre.age);
11     person_change_age(&andre, 30);
12     printf("idade modificada andre %d\n",andre.age);
13
14     Person new_andre = clone_person(&andre);
15
16     person_change_age(&new_andre, 40);
17     printf("idade andre %d\n", andre.age);
18     printf("idade new_andre %d\n", new_andre.age);
19
20     destroy_person(&new_andre);
21     destroy_person(&andre);
22
23     return 0;
24 }
```

```c
1  #include <stdio.h>
2  #include "person.h"
3  #include <stdlib.h>
4  #include <string.h>
5
6  Person new_person(char* name, int age) {
7      // para ilustrar malloc + memcpy
8      size_t n = strlen(name) + 1;
9      char* s = malloc(sizeof(char[n]));
10     memcpy(s, name, n);
11     // ou simplesmente
12     // char* s = strdup(name);
13     return (Person) {
14         .name = s,
15         .age = age,
16     };
17 }
18
19 Person clone_person(Person* p) {
20     return (Person) {
21         .name = strdup(p->name),
22         .age = p->age,
23     };
24 }
```

Makefile

```makefile
1 CC = gcc
2 CFLAGS = -Wall
3
4 program: program.o person.o
5
6 person.o: person.h person.c
7
8 program.o: person.h program.c
9
10 clean:
11     rm -f program *.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
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <strings.h>
4 #include "person.h"
5
6 int main(int argc, char* argv[]){
7
8     Person andre = new_person("Andre", 20);
9
10    printf("idade anterior andre %d\n",andre.age);
11    person_change_age(&andre, 30);
12    printf("idade modificada andre %d\n",andre.age);
13
14    Person new_andre = clone_person(&andre);
15
16    person_change_age(&new_andre, 40);
17    printf("idade andre %d\n", andre.age);
18    printf("idade new_andre %d\n", new_andre.age);
19
20    destroy_person(&new_andre);
21    destroy_person(&andre);
22
23    return 0;
24 }
```

```c
1 #include <stdio.h>
2 #include "person.h"
3 #include <stdlib.h>
4 #include <string.h>
5
6 Person new_person(char* name, int age) {
7     // para ilustrar malloc + memcpy
8     size_t n = strlen(name) + 1;
9     char* s = malloc(sizeof(char[n]));
10    memcpy(s, name, n);
11    // ou simplesmente
12    // char* s = strdup(name);
13    return (Person) {
14        .name = s,
15        .age = age,
16    };
17 }
18
19 Person clone_person(Person* p) {
20    return (Person) {
21        .name = strdup(p->name),
22        .age = p->age,
23    };
24 }
```

Scope

```c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <strings.h>
4 #include "person.h"
5
6 int main(int argc, char* argv[]){
7
8     Person andre = new_person("Andre", 20);
9
10     printf("idade anterior andre %d\n",andre.age);
11     person_change_age(&andre, 30);
12     printf("idade modificada andre %d\n",andre.age);
13
14     Person new_andre = clone_person(&andre);
15
16     person_change_age(&new_andre, 40);
17     printf("idade andre %d\n", andre.age);
18     printf("idade new_andre %d\n", new_andre.age);
19
20     destroy_person(&new_andre);
21     destroy_person(&andre);
22
23     return 0;
24 }
```

```c
30 int person_age(Person* p){
31     return p->age;
32 }
33
34 void person_change_age(Person* p, int age){
35     p->age = age;
36 }
```

Memória Dinâmica

```c
 6 Person new_person(char* name, int age) {
 7     // para ilustrar malloc + memcpy
 8     size_t n = strlen(name) + 1;
 9     char* s = malloc(sizeof(char[n]));
10     memcpy(s, name, n);
11     // ou simplesmente
12     // char* s = strdup(name);
13     return (Person) {
14         .name = s,
15         .age = age,
16     };
17 }
18
19 Person clone_person(Person* p) {
20     return (Person) {
21         .name = strdup(p->name),
22         .age = p->age,
23     };
24 }
25
26 void destroy_person(Person* p) {
27     free(p->name);
28 }
```

Outras Ferramentas

- Man

- gdb, lldb

- Valgrind