

x86-64

Programação genérica

Programação em Sistemas Computacionais

João Pedro Patriarca (joao.patriarca@isel.pt), Gabinete F.0.23 do edifício F

ISEL, ADEETC, LEIC

Agenda

- Ponteiro genérico em C
- Ponteiro para função em C
- Programação genérica em C

Agenda

- Ponteiro genérico em C
- Ponteiro para função em C
- Programação genérica em C


Ponteiro genérico em C

- Um ponteiro guarda sempre um endereço, independentemente do tipo apontado, portanto, a dimensão de um ponteiro é sempre a mesma
- Um ponteiro genérico em C é representado por *void **
- Limitações do tipo *void **
 - Não pode ser desreferenciado
 - Não pode ser envolvido em expressões aritméticas (aritmética de ponteiros)
 - Resumindo, não pode ser envolvido onde seja necessário conhecer a dimensão do tipo apontado
- Para uma programação genérica é preciso envolver a dimensão do tipo apontado

Exemplo com ponteiros genéricos

```
long sum_array_generic1(void * a, int size, int elem_size) {  
    long sum = 0;  
    char * p = (char*)a;  
    while(size-- > 0) {  
        long val = 0;  
        for (int cnt = 0; cnt < elem_size; cnt++)  
            val = val + ((*p++) << (cnt * 8));  
        sum += val;  
    }  
    return sum;  
}
```

sum_array_generic1(..., char) return 10
sum_array_generic1(..., long) return 15



```
void f0_pointer_to_variable() {  
    char a1[] = {1, 2, 3, 4};  
    long r1 = sum_array_generic1(a1, ARRAY_SIZE(a1), sizeof(char));  
    printf("sum_array_generic1(..., char) return %ld\n", r1);  
    long a2[] = {1, 2, 3, 4, 5};  
    long r2 = sum_array_generic1(a2, ARRAY_SIZE(a2), sizeof(long));  
    printf("sum_array_generic1(..., long) return %ld\n", r2);  
}
```

Agenda

- Ponteiro genérico em C
- Ponteiro para função em C
- Programação genérica em C

Ponteiro para função

- Tal como um ponteiro pode guardar o endereço de uma variável, pode, igualmente, guardar o endereço de uma função
 - A afetação do endereço de uma função num ponteiro para função não precisa da utilização do operador &, tal como acontece com uma variável do tipo *array*

```
long get_int(void * pv);  
long (*pfunc)(void *) = get_int;
```

- A chamada a uma função por via de um ponteiro deixa de ser uma chamada direta para passar a ser uma chamada indireta
 - Para chamar indiretamente a função, utiliza-se o nome do ponteiro seguido de abrir parêntesis, argumentos e fechar parêntesis, tal como na chamada normal a uma função

```
int i = 10;  
long v = pfunc(&i); // Traduzido por: call *<reg> ou call *<mem>
```

Exemplo com ponteiro para função

```
long sum_array_generic2(
    void * a, int size,
    int elem_size,
    long (*get_value)(void *))
{
    long sum = 0;
    char * p = (char*)a;
    while(size-- > 0) {
        sum += get_value(p);
        p += elem_size;
    }
    return sum;
}
```

sum_2(..., char) = 10
sum_2(..., long) = 15



```
long get_value_char(void * pv)
{ return *(char*)pv; }
long get_value_long(void * pv)
{ return *(long*)pv; }

void f1_pointer_to_variable_and_function() {
    char a1[] = {1, 2, 3, 4};
    long r1 = sum_array_generic2(
        a1, ARRAY_SIZE(a1), sizeof(char),
        get_value_char);
    printf("sum_2(..., char) = %ld\n", r1);
    long a2[] = {1, 2, 3, 4, 5};
    long r2 = sum_array_generic2(
        a2, ARRAY_SIZE(a2), sizeof(long),
        get_value_long);
    printf("sum_2(..., long) = %ld\n", r2);
}
```


Agenda

- Ponteiro genérico em C
- Ponteiro para função em C
- Programação genérica em C

Exemplo com programação genérica

```
void memswap(void * elem1, void * elem2, int elem_size) {
    char tmp[elem_size];
    memcpy(tmp, elem1, elem_size);
    memcpy(elem1, elem2, elem_size);
    memcpy(elem2, tmp, elem_size);
}

void selection_sort_generic(
    void * a, int size, int elem_size, int (*cmp)(void *, void*))
{
    for (int i = 0; i < size-1; i++) {
        int min_idx = i;
        for (int j = i+1; j < size; j++)
            if (cmp(((char*)a + j*elem_size),
                    ((char*)a + min_idx*elem_size)) < 0)
                min_idx = j;
        if (min_idx != i)
            memswap((char*)a + i*elem_size,
                    (char*)a + min_idx*elem_size,
                    elem_size);
    }
}
```

Exercício

- Traduza para *assembly* x86-64 a função *selection_sort_generic* definida no slide anterior