Block

局部变量
Int localVar = 1;

直接取值
struct __main_block_impl_0 持有
int localVar;

局部指针变量
int* i = &localVar;

直接取值(值就是地址)
struct __main_block_impl_0 持有
int *i;

__block修饰的局部变量
__block localBlockVar = 1;

转换成结构体指针
```
struct __Block_byref_localBlockVar_0
{
    void *__isa;

__Block_byref_localBlockVar_0 *__forwarding;
    int __flags;
    int __size;
    int localBlockVar1; =1;
   };
struct __main_block_impl_0 持有
__Block_byref_localBlockVar_0 *localBlockVar; // by ref
```

局部静态变量
static int localStaticVar = 1;

转换成指针
int *localStaticVar;(取localStaticVar地址,即 int *x =&localStaticVar)
struct __main_block_impl_0 持有
int *localStaticVar;

全局静态变量
static int globalStaticVar = 1;

直接使用(不会转化,不会持有,不会copy,内存在全局区,不存在提前释放)

全局变量
int globalVar = 1;

直接使用(不会转化,不会持有,不会copy,内存在全局区,不存在提前释放)

指针变量
NSObject* objc = [[NSObject alloc]init];

初始化
__main_block_impl_0(传入各种参数)

```
struct __main_block_impl_0 {
      struct __block_impl impl;//block接口实现
      struct __main_block_desc_0* Desc;//block内存管理
      //此时,以下
      int localVar;
      int localVarTest;
      int *s;

      //------静态 局部变量 !转换! 成 (int*) 类型------
      //明明定义的是static int localStaticVar = 2;
      //也就是说,int *localStaticVar = &localStaticVar;
      int *localStaticVar;// why??
      //------------

      //------__block修饰的 局部变量 !转换! 成(__Block_byref_localBlockVar1_0*) 类型------
      __Block_byref_localBlockVar1_0 *localBlockVar1; //by ref
      __Block_byref_localBlockVar2_1 *localBlockVar2; //by ref
      //------------

      //初始化__main_block_impl_0 传入各种参数
      __main_block_impl_0(void *fp,
                          struct __main_block_desc_0 *desc,
                          int _localVar,
                          int* _localStaticVar,
                          int _localVarTest,
                          int* _s,
                          __Block_byref_localBlockVar1_0 *_localBlockVar1,
                          __Block_byref_localBlockVar2_1 *_localBlockVar2,
                          int flags=0
      ) : localVar(_localVar),
          localStaticVar(_localStaticVar),
          localVarTest(_localVarTest),
          s(_s),
          localBlockVar1(_localBlockVar1->__forwarding),
          localBlockVar2(_localBlockVar2->__forwarding) {
                impl.isa = &_NSConcreteStackBlock;//栈
                impl.Flags = flags;
                impl.FuncPtr = fp;
                Desc = desc;
          }//赋值
    };
```

执行[block copy]
__main_block_impl_0 copy到heap

__main_block_impl_0 内的参数:
局部基本数据类型: copy新地址(heap上)
局部指针类型: copy新指针(heap上)指向原地址

__main_block_impl_0 内的参数:
__block修饰的局部变量:
__Block_byref_localBlockVar_0 *localBlockVar1;
copy到堆上

```
    static void __main_block_copy_0(struct __main_block_impl_0*dst,struct __main_block_impl_0*src) {
        _Block_object_assign((void*)&dst->localBlockVar1, (void*)src->localBlockVar1, 8/*BLOCK_FIELD_IS_BYREF*/);

                //(8参数)用于区分对象和__block变量
                //BLOCK_FIELD_IS_BYREF  代表 __block变量
                //BLOCK_FIELD_IS_OBJECT 代表 对象

        _Block_object_assign((void*)&dst->localBlockVar2, (void*)src->localBlockVar2, 8/*BLOCK_FIELD_IS_BYREF*/);
    }
```
_Block_object_assign 会根据weak strong copy…修饰符,执行对应的操作

不执行copy操作
__main_block_impl_0 仍然在stack内部参数 也在stack
//参数和block都会被提前释放

调用Block()时,调用函数
static void __main_block_func_0(传入__main_block_impl_0)

```
static void __main_block_func_0(struct __main_block_impl_0 *__cself) {
        //通过__cself指针
        //找到__main_block_impl_0内部的
        //__Block_byref_localBlockVar1_0* localBlockVar1or2(指针变量)
      __Block_byref_localBlockVar1_0 *localBlockVar1 = __cself->localBlockVar1; // bound by ref
      __Block_byref_localBlockVar2_1 *localBlockVar2 = __cself->localBlockVar2; // bound by ref
        //localVar(值)
      int localVar = __cself->localVar; // bound by copy
        //localStaticVar(指针变量)
      int *localStaticVar = __cself->localStaticVar; // bound by copy
        //localVarTest(值)
      int localVarTest = _cself->localVarTest; // bound by copy
        //s(指针变量)
      int *s = __cself->s; // bound by copy

            printf("---------block内部:---------");

            printf("全局变量:%d",globalVar);
            printf("全局变量地址:%p",&globalVar);
            printf("全局静态变量:%d",globalStaticVar);
            printf("全局静态变量地址:%p",&globalStaticVar);

            printf("局部变量:%d",localVar);
            printf("局部变量地址:%p",&localVar);
            printf("局部静态变量:%d",(*localStaticVar));
            printf("局部静态变量地址:%p",&(*localStaticVar));

            printf("__block修饰的局部变量1:%d",(localBlockVar1->__forwarding->localBlockVar1));
            printf("__block修饰的局部变量1地址:%p",&(localBlockVar1->__forwarding->localBlockVar1));
            printf("__block修饰的局部变量2:%d",(localBlockVar2->__forwarding->localBlockVar2));
            printf("__block修饰的局部变量地址2:%p",&(localBlockVar2->__forwarding->localBlockVar2));

            printf("s:%d",*s);
            printf("s地址:%p",&s);
        }
```

//------clang后的 main函数-------

```
int main () {
    //局部变量部分
    printf("---------block外部:---------");
    int localVar = 1;
    printf("局部变量地址:%p",&localVar);
    static int localStaticVar = 2;
    printf("局部静态变量地址:%p",&localStaticVar);
    int localVarTest = 100;
    int* s = &localVarTest;
    printf("s地址:%p",&s);
```

```
    //__block修饰的局部变量部分
    //__block的作用:是为了内部外部局部变量同步,外部可以修改内部截获的变量
    //通过__Block_byref_localBlockVar1_0,__forwarding访问变量
     //从而做到 外部可以修改内部截获变量
    __attribute__((__blocks__(byref))) __Block_byref_localBlockVar1_0 localBlockVar1 = {
        (void*)0,
        (__Block_byref_localBlockVar1_0 *)&localBlockVar1,
        0,
        sizeof(__Block_byref_localBlockVar1_0),
        3
    };
    printf("__block修饰的局部变量1地址:%p",&(localBlockVar1.__forwarding->localBlockVar1));

    __attribute__((__blocks__(byref))) __Block_byref_localBlockVar2_1 localBlockVar2 = {
        (void*)0,
        (__Block_byref_localBlockVar2_1 *)&localBlockVar2,
        0,
        sizeof(__Block_byref_localBlockVar2_1),
        4
    };
    printf("__block修饰的局部变量地址2:%p",&(localBlockVar2.__forwarding->localBlockVar2));
```

```
    //初始化block
    void (*v)(void) = ((void (*)())&__main_block_impl_0(
                (void *)__main_block_func_0,
                &__main_block_desc_0_DATA,
                localVar,
                &localStaticVar, l
                ocalVarTest,
                s,
                (__Block_byref_localBlockVar1_0 *)&localBlockVar1,
                (__Block_byref_localBlockVar2_1 *)&localBlockVar2,
                570425344)
    );
    //上面函数简写
    __main_block_impl_0 tmp = __main_block_impl_0(
                                    (void *)__main_block_func_0,
                                    &__main_block_desc_0_DATA,
                                    localVar,
                                    &localStaticVar,
                                    localVarTest,
                                    s,
                                    (__Block_byref_localBlockVar1_0 *)&localBlockVar1,
                                    (__Block_byref_localBlockVar2_1 *)&localBlockVar2,
                                    570425344);
     struct __main_block_impl_0 *blk = &tmp;
    ///
```

```
    //block外部 对变量修改
    globalVar++;
    globalStaticVar++;

    localVar++;
    localStaticVar++;

    //通过__forwarding访问
    (localBlockVar1.__forwarding->localBlockVar1)++; //保证了内外同步
    (localBlockVar2.__forwarding->localBlockVar2)++;

    (*s)++;
```

```
    //执行block内部函数
    ((void (*)(__block_impl *))((__block_impl *)v)->FuncPtr)((__block_impl *)v);
    //上面函数简写
    ((*blk->impl)->FuncPrt)(blk);
    ///
    return 0;
}
```