Blocks

WG14
April 20, 2010

Blaine Garst

blaine@apple.com

Your First Blocks

Multilingual

Il Cocco di Mamma

Blocks for Adults

Cuboro, handmade in Switzerland

Blocks for C, Objective-C, and C++

```
void forEachLine(const char *p, void (^b)(const char *)) {
  FILE *fp = fopen(p); if (!fp) return;
  char buffer[8196];
  while (fgets(buffer, 8196, fp))
    b(fp);
  fclose(fp);
}
```

```
void foo() {
   __block int acheCounter = 0;
   forEachLine("/tmp/foo", ^(const char *line) {
    if (strstr("ake")) ++acheCounter;
   });
}
```

Blocks for C, Objective-C, and C++

```
NSString *x = [NSString stringWithContentsOfFile:...];
__block int acheCounter = 0;

[x enumerateLinesUsingBlock:^(NSString *line, BOOL *stop) {
    if ([line containsString:@"ake"])
        ++acheCounter;
}];
```

Existing qsort_r

```
typedef struct {
  int col, len;
  int col2, len2;
} twoway_t;
```

sort strings with two fixed column positions & lengths

```
int compareTwoWay(char *l, char *r, twoway_t *tw) {
  int result = strncmp(l+tw->col, r+tw->col, tw->len);
  if (result == 0)
    result = strncmp(l+tw->col2, r+tw->col2, tw->len2);
  return result;
}
```

```
int col = ..., len = ..., col2 = ..., len2 = ...;
...
twoway_t ctx = { col, len, col2, len2 };
qsort_r(lines, nitems, sizeof(char *), &ctx, compareTwoWay);
```

qsort_b: qsort with Blocks

sort strings with two fixed column positions & lengths

```
int col = ..., len = ..., col2 = ..., len2 = ...;
qsort_b(lines, nitems, sizeof(char *), ^(char *l, char *r) {
  int result = strncmp(l+col, r+col, len);
  if (result == 0)
    result = strncmp(l+col2, r+col2, len2);
  return result;
});
```

Callback with void * info Example

```
void myCallBackFunction (
                CFNetServiceBrowserRef browser,
                CFOptionFlags flags,
                CFTypeRef domainOrService,
                CFStreamError* error,
                void* info)
{
    struct MyInfo *myInfo = (struct myInfo *)info;
    ...; // do your thing
}
```

```
....
CFNetServiceBrowserRef myBrowser = CFNetServiceBrowserCreate (
    NULL, myCallBackFunction, myWrappedDataPtr);
```

Callback with void * info Example

```
void myCallBackFunction (
                CFNetServiceBrowserRef browser,
                CFOptionFlags flags,
                CFTypeRef domainOrService,
                CFStreamError* error,
                void* info)
{
    struct MyInfo *myInfo = (struct myInfo *)info;
    ...; // do your thing
}
```

```
....
CFNetServiceBrowserRef myBrowser = CFNetServiceBrowserCreate (
    NULL, myCallBackFunction, myWrappedDataPtr);
```

Callback with void * info Example

```
const void *myRetain(const void *info) {
    // lock and bump a refcount in info?  malloc?
}
void myRelease(const void *info) {
    // lock and decr a refcount or just free?
}
```

```
struct CFNetServiceClientContext *myWrappedDataPtr =
                                CFAllocatorAllocate(NULL, ...);
myWrappedDataPtr->version = 0;
myWrappedDataPtr->retain = myRetain;
myWrappedDataPtr->release = myRelease;
myWrappedDataPtr->info = myInfo;
....
```

Imagine This Callback Using a Block

```
struct MyInfo myInfo = { ... };
CFNetServiceBrowserRef myBrowser = CFNetServiceBrowserCreate_b(
    NULL,
    ^ (CFNetServiceBrowserRef browser, CFOptionFlags flags,
                CFTypeRef domainOrService, CFStreamError* error) {
      // do your thing
      ... myInfo.xxxx ....
});
```

Block Syntax

Block Abstract Type Specifier

- Like a function pointer, a Block has a return type and an optional list of parameter types

```
returntype (^)(argtype1, argtype2)
```

- This is used when declaring Objective-C parameters

```
- methodWithBlock:(returntype (^)(argtype1, argtype2)) block;
```

- Block pointer variables must use C declaration style

```
returntype (^blockVariable)(argtype1, argtype2);
```

- An array of Block pointers

```
returntype (^blockArray[4])(argtype1, argtype2);
```

More Fun with Declarators

- A function (taking an int) returning a Block

```
returntype (^func(int farg))(argtype1, argtype2) {...}
```

- A pointer to a function (taking an int) returning a Block

```
returntype (^(*pfunc)(int farg))(argtype1, argtype2);
```

- A pointer variable to a Block (taking an int) returning a Block

```
returntype (^(^block)(int barg))(argtype1, argtype2);
```

Block Expression Literal Syntax

- Return type is inferred from return statement

```
^(int arg1, char arg2) { return arg1+arg2; } //  int (^)(int, char)
^(int arg1, char arg2) { arg1+arg2; }        // void (^)(int, char)
```

- Block literals with a void argument can avoid the void

```
^(void){ sleep(3); }  // no args
^{ sleep(3); }        // same thing
```

- __block storage for mutable items

```
__block int x, y, z;
void (^sleeper)(void) = ^{
  sleep(++x + ++y + ++z);
}
```

Block Expression Syntax

Automatic local variables imported as const

```
void foo() {
   int y;
   void (^incr)(void) = ^{
     y += 4;     // ERROR: y imported as const
   };
}
```

Block Expression Syntax

Static file, global and extern variables work normally

```
static int StaticX;
int GlobalY;
extern int ExternZ;

void bar() {
  static int localW;
  void (^globIncr)(void) = ^{
     StaticX += 3;
     GlobalY += 4;
     ExternZ += 5;
     LocalW += 6;
  };
```

Block Literals Are Stack Based Objects

- Don’t return them directly (compiler warns)

```
return ^{ printf("hello\n"); }; // will crash!!
```

- This counts as directly

```
void (^blockVar)(void) = ^{ printf("hello\n"); };
return blockVar;                    // will crash!!!
```

Global Blocks (optimization)

```
#include <Foundation/Foundation.h>

int (^countValues)(NSArray *) = ^(NSArray *a) {
    int result = 0;
    for (NSNumber *numb in a) {
       result += [numb intValue];
    }
    return result;
};
```

Apple Block ABI

SnowLeopard ABI for block object

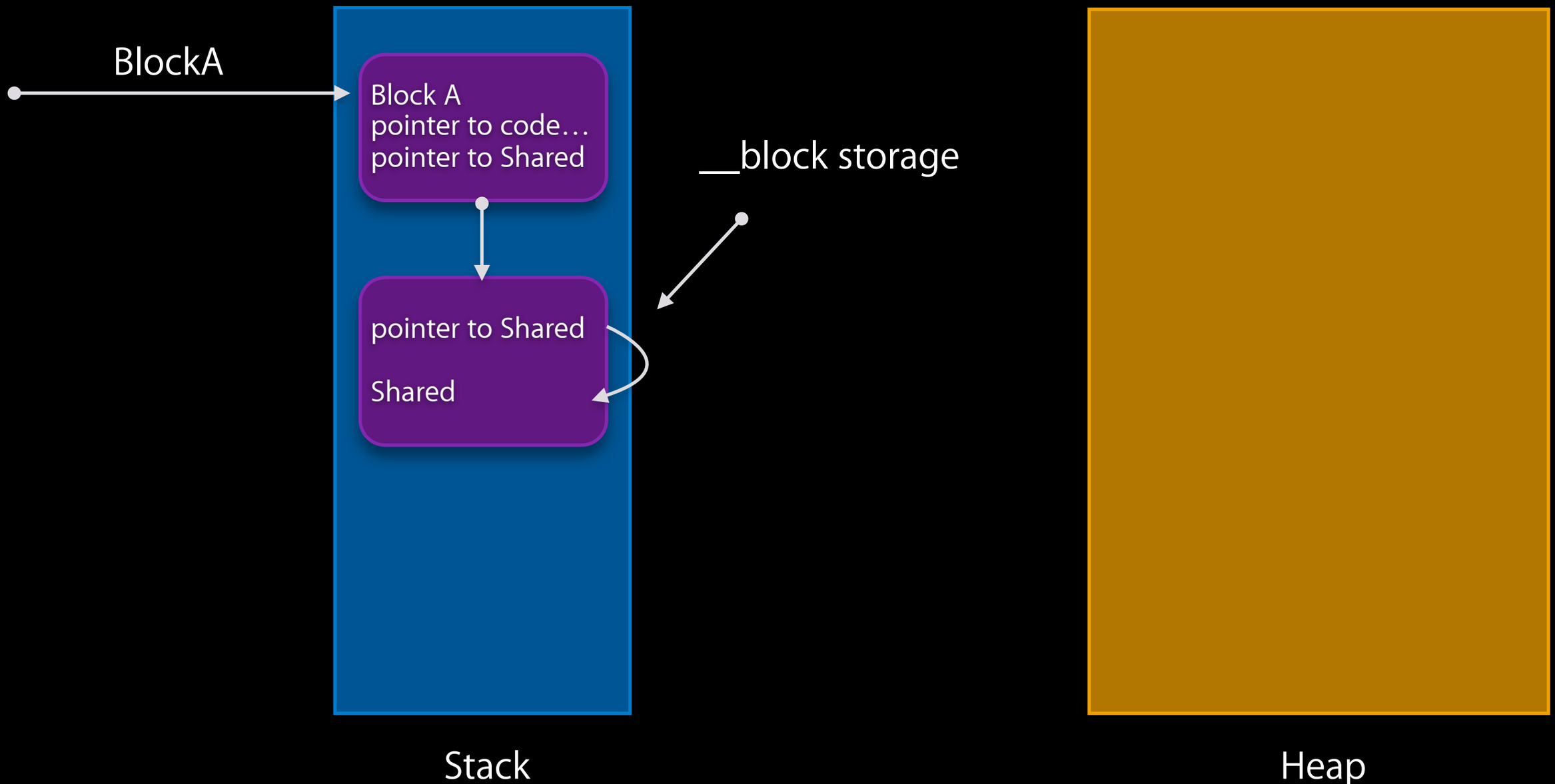
```
struct Block_layout {
    void *isa;
    int flags;
    int reserved;
    void (*invoke)(void *, ...);
    struct Block_descriptor *descriptor;
    /* const variables. */
};
```

```
struct Block_descriptor {
    unsigned long int reserved;
    unsigned long int size;
    void (*copy)(void *dst, void *src);
    void (*dispose)(void *);
};
```

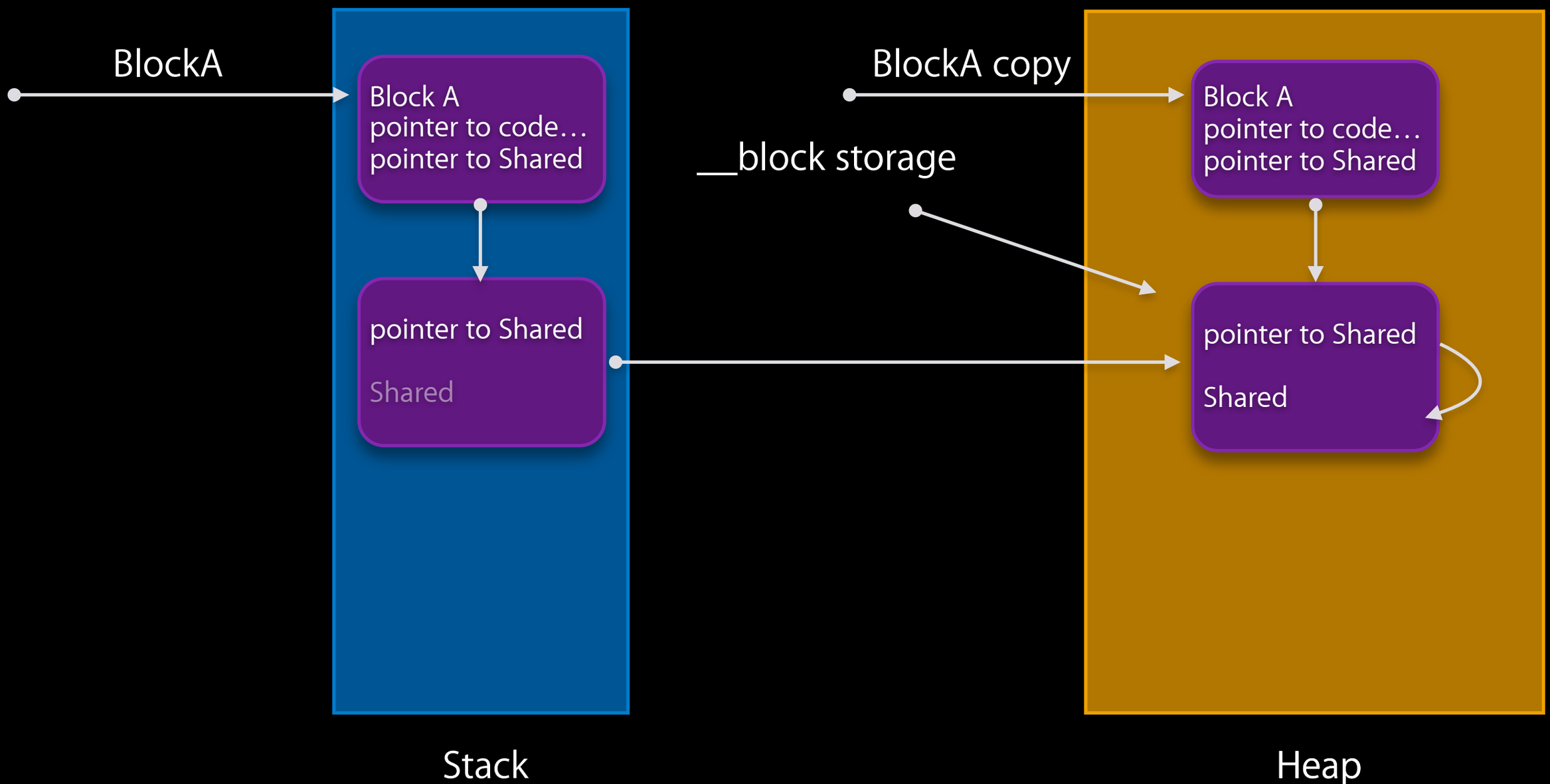
SnowLeopard ABI for __block object

```
struct Block_mutable {
    void *isa;
    struct Block_mutable *forwarding;
    int flags; /* refcount; */
    int size;
    void (*copy)(struct Block_mutable *dst, struct Block_mutable *src);
    void (*free)(struct Block_mutable *);
    /* mutable variable goes here */
};
```

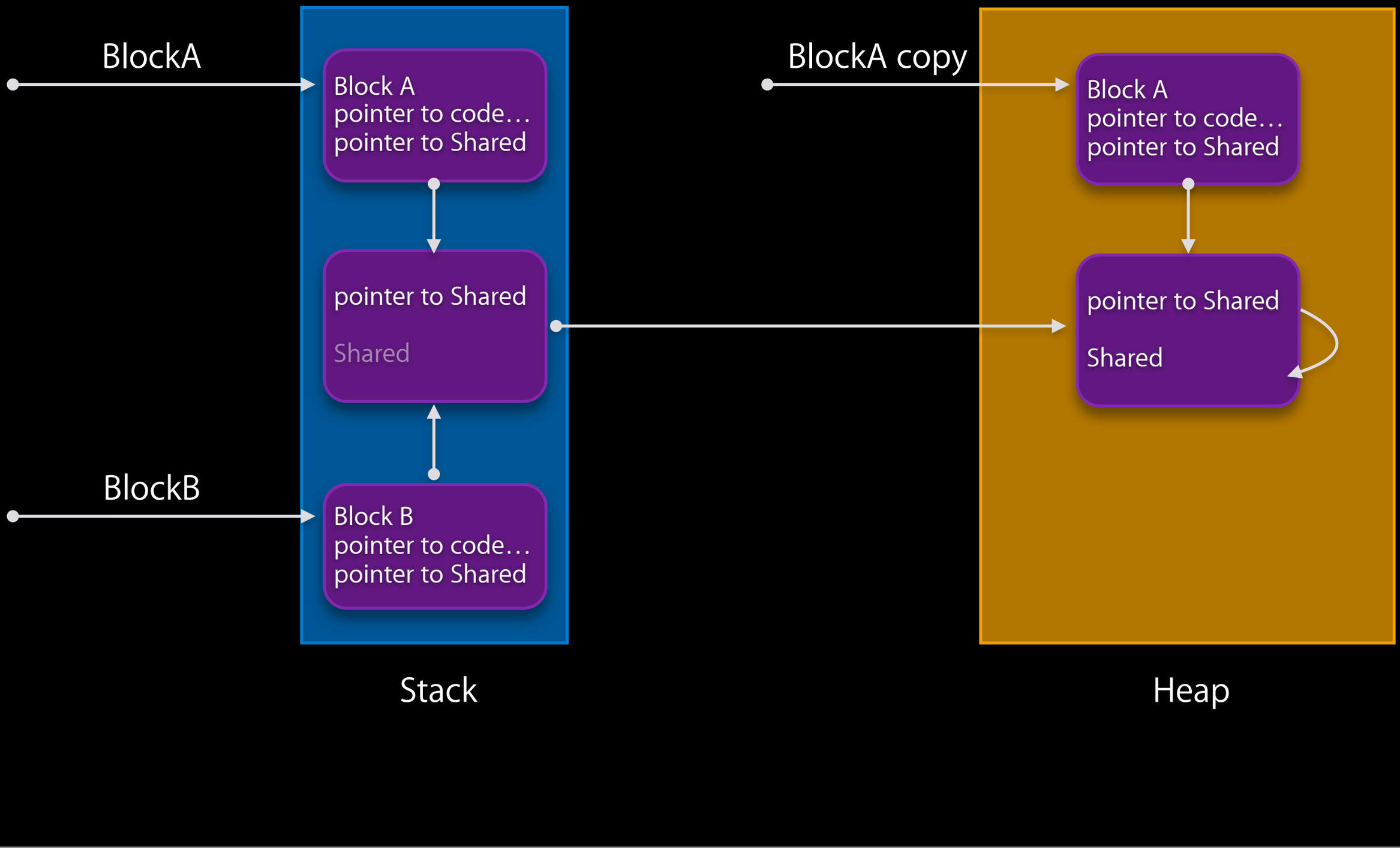
Before Block_copy



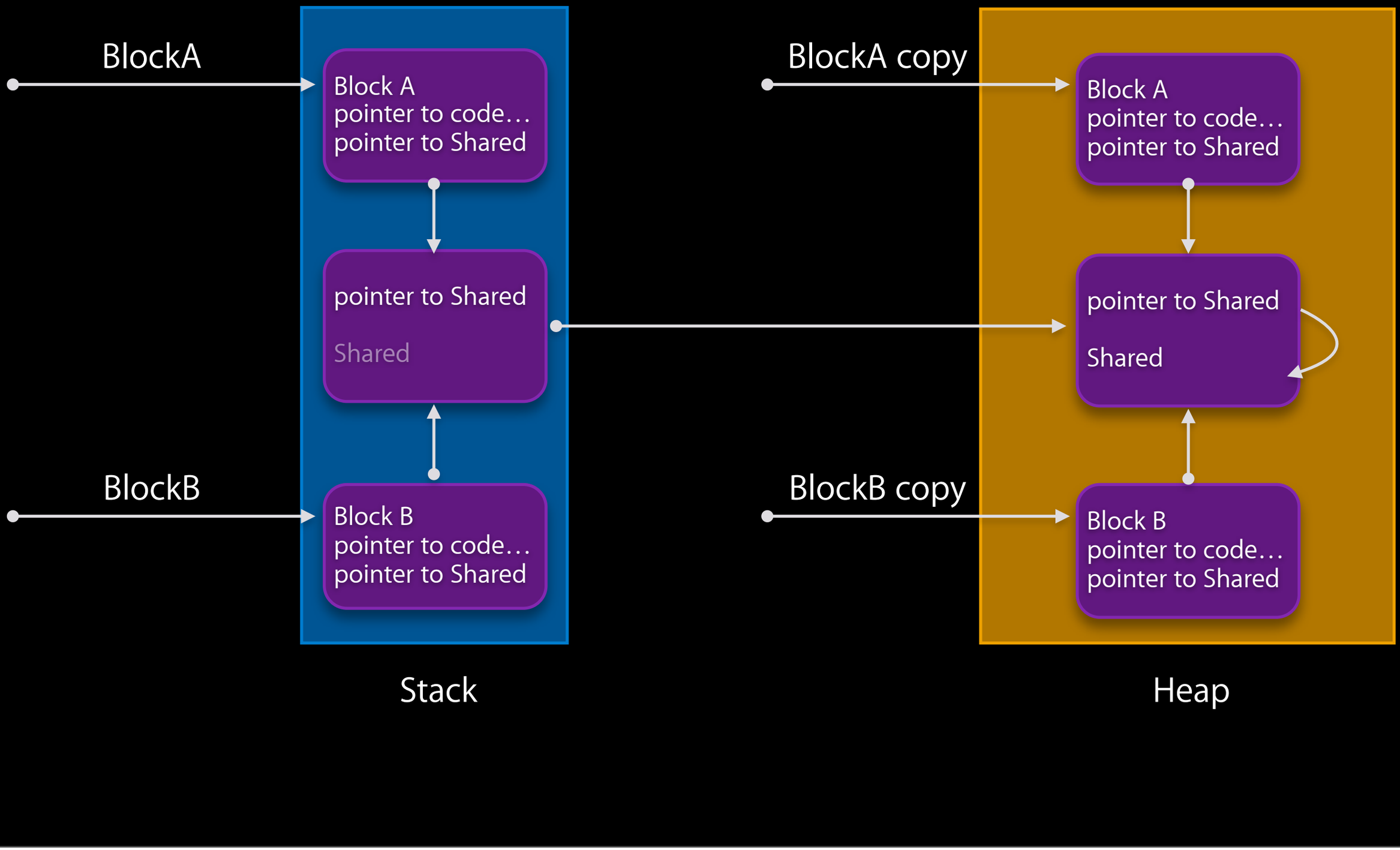
After Block_copy



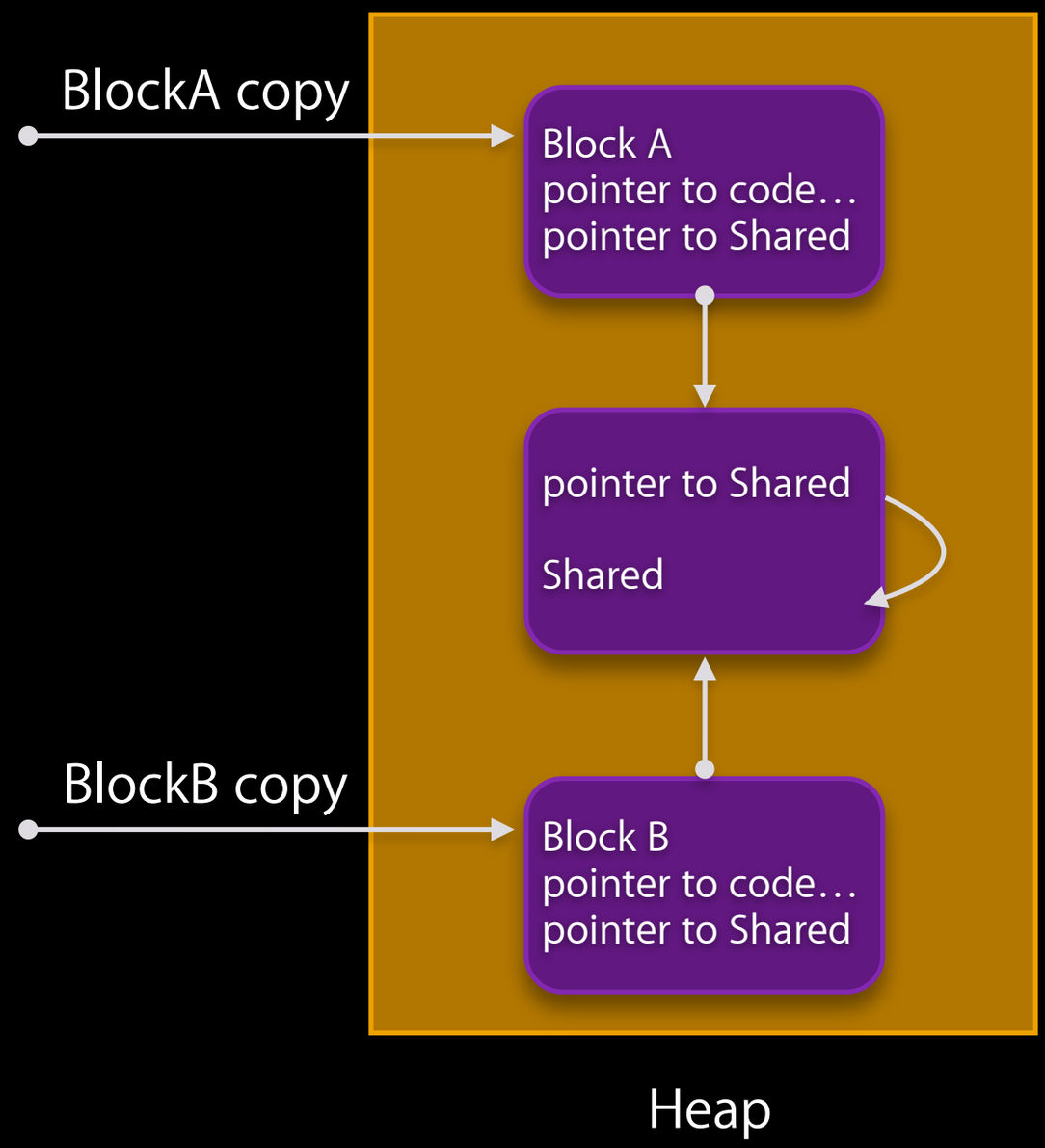
BlockB Sees Correct Shared Variable



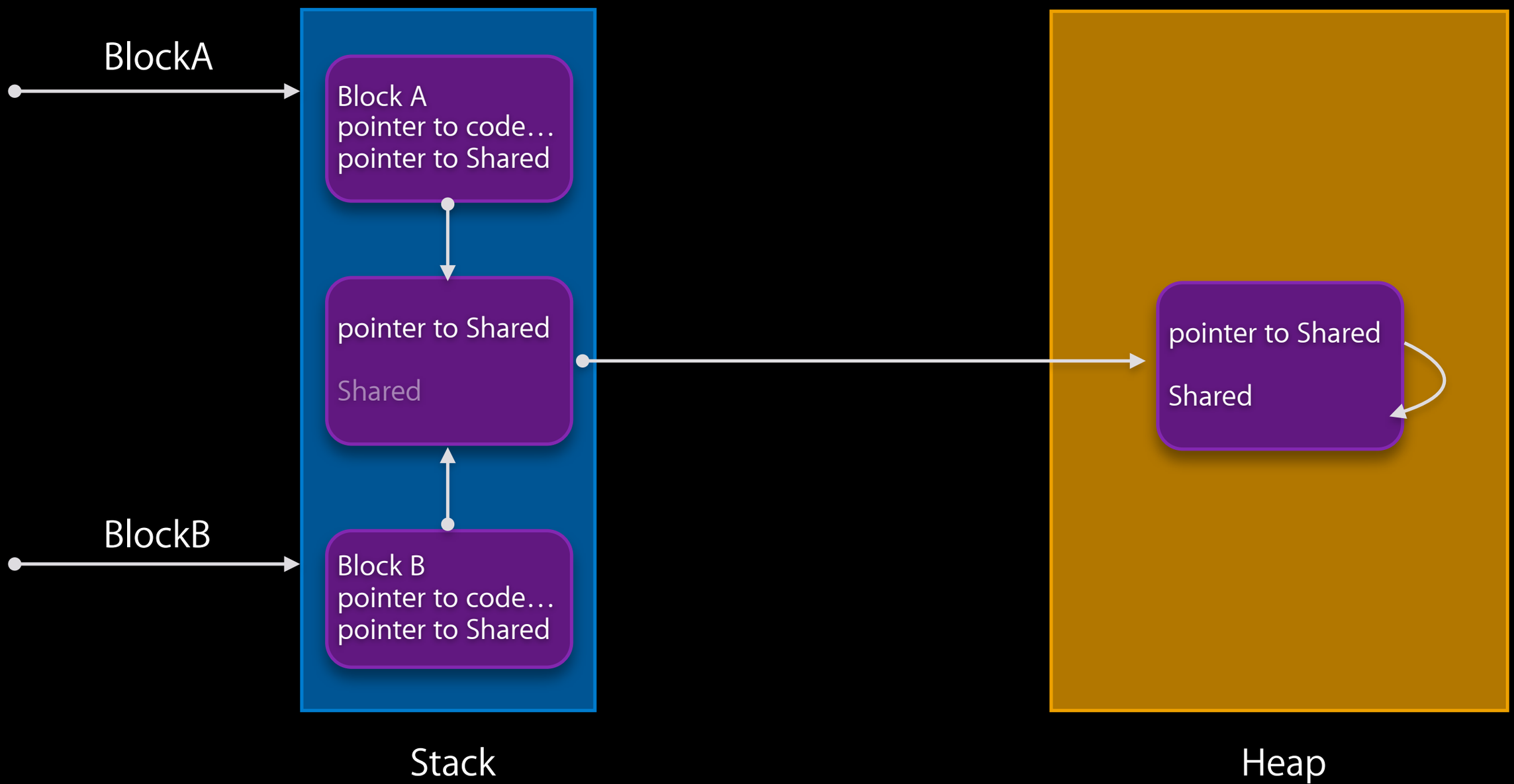
Even After a BlockB Copy



Block Copies Work When Stack Pops



Or if the Copies Go Away First



Resources

- blaine@apple.com
- Overview: N1370.pdf
- Proposal: N1451.pdf
- LLVM spec and ABI: ("cfe" == Clang Front End)
- http://llvm.org.svn/llvm-project/cfe/trunk/docs
  - BlockLanguage.txt
  - Block-ABI-Apple.txt
- LLVM runtime:
  - http://llvm.org/svn/llvm-project/compiler-rt/trunk
    - BlockRuntime
      - runtime.c, data.c
      - Block.h, Block_private.h

License

```
/*
 * runtime.c
 *
 * Copyright 2008-2009 Apple, Inc. Permission is hereby granted, free of charge,
 * to any person obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without restriction,
 * including without limitation the rights to use, copy, modify, merge, publish,
 * distribute, sublicense, and/or sell copies of the Software, and to permit
 * persons to whom the Software is furnished to do so, subject to the following
 * conditions:
 *
* The above copyright notice and this permission notice shall be included in
* all copies or substantial portions of the Software.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*
*/
```