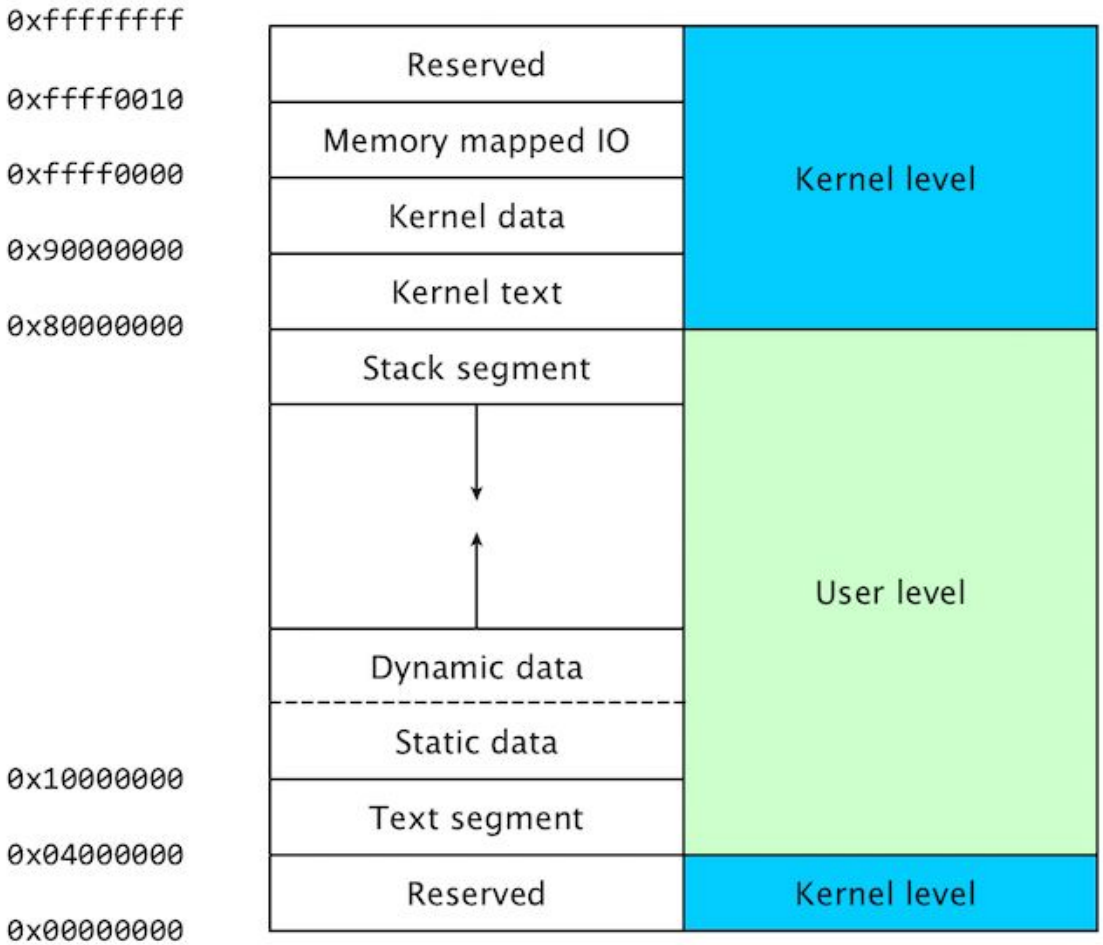
A heap, a stack, a bottle and a rack

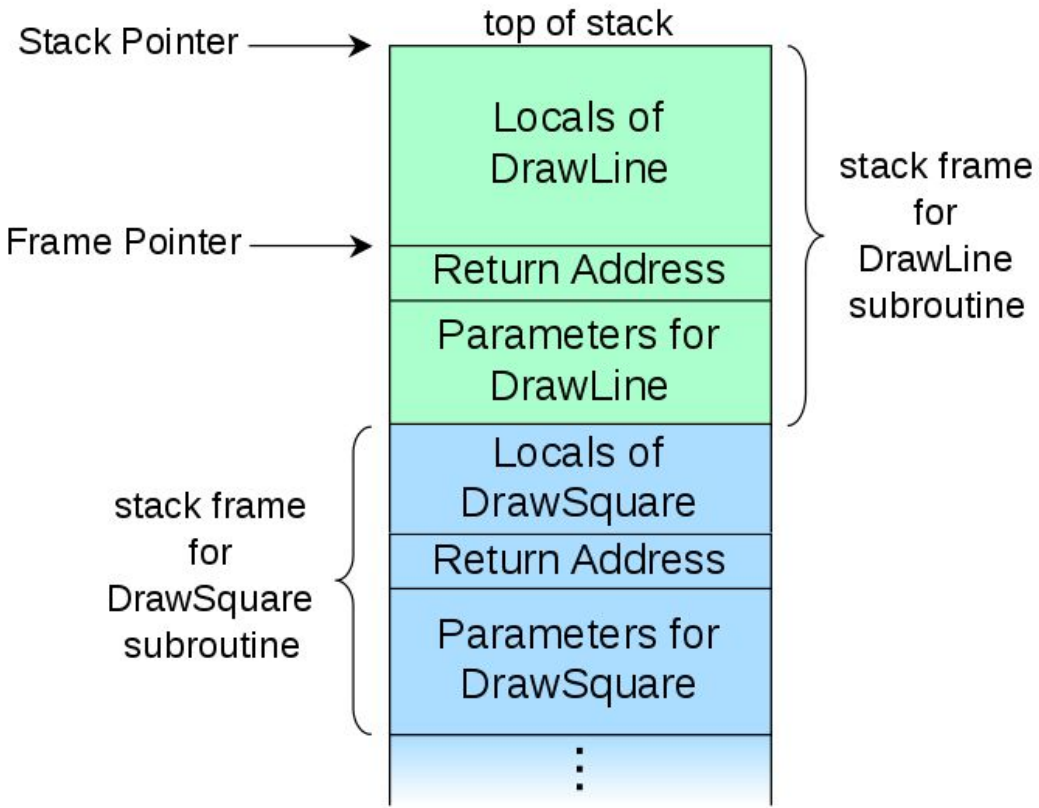

0xffffffff
0xffff0010
0xffff0000
0x90000000
0x80000000
0x10000000
0x04000000
0x00000000
Reserved
Memory mapped IO
Kernel data
Kernel text
Stack segment
Dynamic data
Static data
Text segment
Reserved
Kernel level
User level
Kernel level

/proc/<pid>/maps

```
562194b28000-562194b29000 r-xp 00000000 08:02 4719974                    /home/csd/test-p
562194d28000-562194d29000 r--p 00000000 08:02 4719974                    /home/csd/test-p
562194d29000-562194d2a000 rw-p 00001000 08:02 4719974                    /home/csd/test-p
5621961ea000-56219620b000 rw-p 00000000 00:00 0                          [heap]
7f6065546000-7f606572d000 r-xp 00000000 08:02 1966852                    /lib/x86_64-linux-gnu/libc-2.27.so
7f606572d000-7f606592d000 ---p 001e7000 08:02 1966852                    /lib/x86_64-linux-gnu/libc-2.27.so
7f606592d000-7f6065931000 r--p 001e7000 08:02 1966852                    /lib/x86_64-linux-gnu/libc-2.27.so
7f6065931000-7f6065933000 rw-p 001eb000 08:02 1966852                    /lib/x86_64-linux-gnu/libc-2.27.so
7f6065933000-7f6065937000 rw-p 00000000 00:00 0
7f6065937000-7f606595e000 r-xp 00000000 08:02 1966840                    /lib/x86_64-linux-gnu/ld-2.27.so
7f6065b50000-7f6065b52000 rw-p 00000000 00:00 0
7f6065b5e000-7f6065b5f000 r--p 00027000 08:02 1966840                    /lib/x86_64-linux-gnu/ld-2.27.so
7f6065b5f000-7f6065b60000 rw-p 00028000 08:02 1966840                    /lib/x86_64-linux-gnu/ld-2.27.so
7f6065b60000-7f6065b61000 rw-p 00000000 00:00 0
7ffc70711000-7ffc70732000 rw-p 00000000 00:00 0                          [stack]
7ffc707d8000-7ffc707db000 r--p 00000000 00:00 0                          [vvar]
7ffc707db000-7ffc707dd000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

The Stack

- The working memory.



“Canary Birds” (and other stuff on the stack)

- Can be hard to identify all objects on the stack
- Canary birds: To ensure nothing is broken
- Other scap data: To align memory
- Can help to optimize compile (-O) if you want to identify all items

The heap

- *malloc*
- When we need more permanent data
- Called “dynamic data” in previous figure

(S)heap

Exam question 1

What is done in the procedure below and where should gurka be allocated?
Why? Complete the code so that gurka is allocated space.

```
void tomat(int *a, int *b) {
  // allocate room for gurka


  gurka = *a;
  *a = *b;
  *b = gurka;
}
```

Exam question 2

What is done in the procedure below and where should `gurka` be allocated? Why? Complete the code so that `gurka` is allocated space.

```
int *tomat(int *a, int *b) {
  // allocate room for gurka


  *gurka = *a + *b;
  return gurka;
}
```

In the code below we have allocated three arrays where one is on the heap, which array and why is it allocated on the heap and not on the stack?

```
#include <stdlib.h>
#include <stdio.h>

#define MAX 4

int h[MAX];

int *foo(int *a, int *b, int s) {

  int *r = malloc(s * sizeof(int));

  for(int i = 0; i < s; i++) {
     r[i] = a[i]+b[i];
  }
  return r;
}
```

```
int main() {
  int f[MAX];

  for(int i = 0; i < MAX; i++) {
    f[i] = i;
    h[i] = i*10;
  }

  int *g = foo(f, h, 4);

  printf("a[2] + b[2] is %d\n", g[2]);

  return 0;
}
```

1.2 memory map [2 points]

Below is a, somewhat shortened, printout of a memory mapping of a running process. Briefly describe the role of each segment marked with ???.

```
> cat /proc/13896/maps

00400000-00401000 r-xp 00000000 08:01 1723260            .../gurka ???
00600000-00601000 r--p 00000000 08:01 1723260            .../gurka ???
00601000-00602000 rw-p 00001000 08:01 1723260            .../gurka ???
022fa000-0231b000 rw-p 00000000 00:00 0                  [???]
7f6683423000-7f66835e2000 r-xp 00000000 08:01 3149003    .../libc-2.23.so ???
    :
7ffd60600000-7ffd60621000 rw-p 00000000 00:00 0          [???]
7ffd60648000-7ffd6064a000 r--p 00000000 00:00 0          [vvar]
7ffd6064a000-7ffd6064c000 r-xp 00000000 00:00 0          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0  [vsyscall]
```

```
00400000-00401000 r-xp 00000000 08:01 1723260                    .../gurka ???
00600000-00601000 r--p 00000000 08:01 1723260                    .../gurka ???
00601000-00602000 rw-p 00001000 08:01 1723260                    .../gurka ???
022fa000-0231b000 rw-p 00000000 00:00 0                          [???]
7f6683423000-7f66835e2000 r-xp 00000000 08:01 3149003            .../libc-2.23.so ???
    :
7ffd60600000-7ffd60621000 rw-p 00000000 00:00 0                  [???]
7ffd60648000-7ffd6064a000 r--p 00000000 00:00 0                  [vvar]
7ffd6064a000-7ffd6064c000 r-xp 00000000 00:00 0                  [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0          [vsyscall]
```

Answer:
The first three segments are: **code**, **read-only data** and **global data** for the running process gurka.
Then there is a segment for the **heap**.
The segment marked with lib-2.23.so is a **shared library**.
In the uppermost region we nd the segment of the **stack**.

It is not completely defined what will happen if we run the code below. What is it that we do wrong and what could a possible effect be?

```
int main() {

   char *heap = malloc(20);
   *heap = 0x61;
   printf("heap pointing to: 0x%x\n", *heap);
   free(heap);

   char *foo = malloc(20);
   *foo = 0x62;
   printf("foo pointing to: 0x%x\n", *foo);

   *heap = 0x63;
   printf("foo pointing to: 0x%x\n", *foo);

   return 0;
}
```

Below we see a program that will print the content of the stack.

```
void zot(unsigned long *stop, int a1, int a2, int a3, int a4, int a5, int a6) {
  unsigned long r = 0x456;
  unsigned long *i;
  for(i = &r;  i <= stop; i++){
    printf("%p      0x%lx\n", i, *i);
  }
}

int main() {
  unsigned long p = 0x123;

  zot(&p,1,2,3,4,5,6);
 back:
  printf("  back: %p \n", &&back);
  return 0;
}
```

When executed we see the following print out. Describe the values indicated with arrows (<--).

```
0x7ffeb3331f58      0x456
0x7ffeb3331f60      0x7ffeb3331f60  <-- ??
0x7ffeb3331f68      0x3a7dbfad7df4b100
0x7ffeb3331f70      0x7ffeb3331fa0
0x7ffeb3331f78      0x400663    <-- ??
0x7ffeb3331f80      0x6   <-- ??
0x7ffeb3331f88      0x4004a0
0x7ffeb3331f90      0x123
  back: 0x400667
```

You have written the program below to examine what is on the stack.

```
void zot(unsigned long *stop ) {
  unsigned long r = 0x3;
  unsigned long *i;
  for(i = &r;  i <= stop; i++){ printf("%p          0x%lx\n", i, *i);  }
}

void foo(unsigned long *stop ) {
  unsigned long q = 0x2;
  zot(stop);
}

int main() {
  unsigned long p = 0x1;
  foo(&p);
 back:
  printf("  p: %p \n", &p);
  printf("  back: %p \n", &&back);
  return 0;
}
```

```
0x7ffca03d1748      0x3
0x7ffca03d1750      0x7ffca03d1750
0x7ffca03d1758      0xb93d7906926a7d00
0x7ffca03d1760      0x7ffca03d1790        <-------
0x7ffca03d1768      0x55cdac31d78c        <-------
0x7ffca03d1770      0x7ffca03d17d8
0x7ffca03d1778      0x7ffca03d17b0
0x7ffca03d1780      0x1
0x7ffca03d1788      0x2
0x7ffca03d1790      0x7ffca03d17c0
0x7ffca03d1798      0x55cdac31d7c2
0x7ffca03d17a0      0x55cdac31d810
0x7ffca03d17a8      0x12acac31d5f0
0x7ffca03d17b0      0x1
  p: 0x7ffca03d17b0
  back: 0x55cdac31d7c2
```

Relevant Sections from the Book

For reading at your own time

- Chapter 13 - The Abstraction: Address Spaces
- Chapter 14 - Interlude: Memory API