Dynamic Memory

Allocation in C

```
#include <stdlib.h>

void *malloc(size_t size);
void free(void* ptr);
```

Allocate an Array

```
int *pArray;
const int SIZE = 1024;

pArray   =              malloc(




free(
```

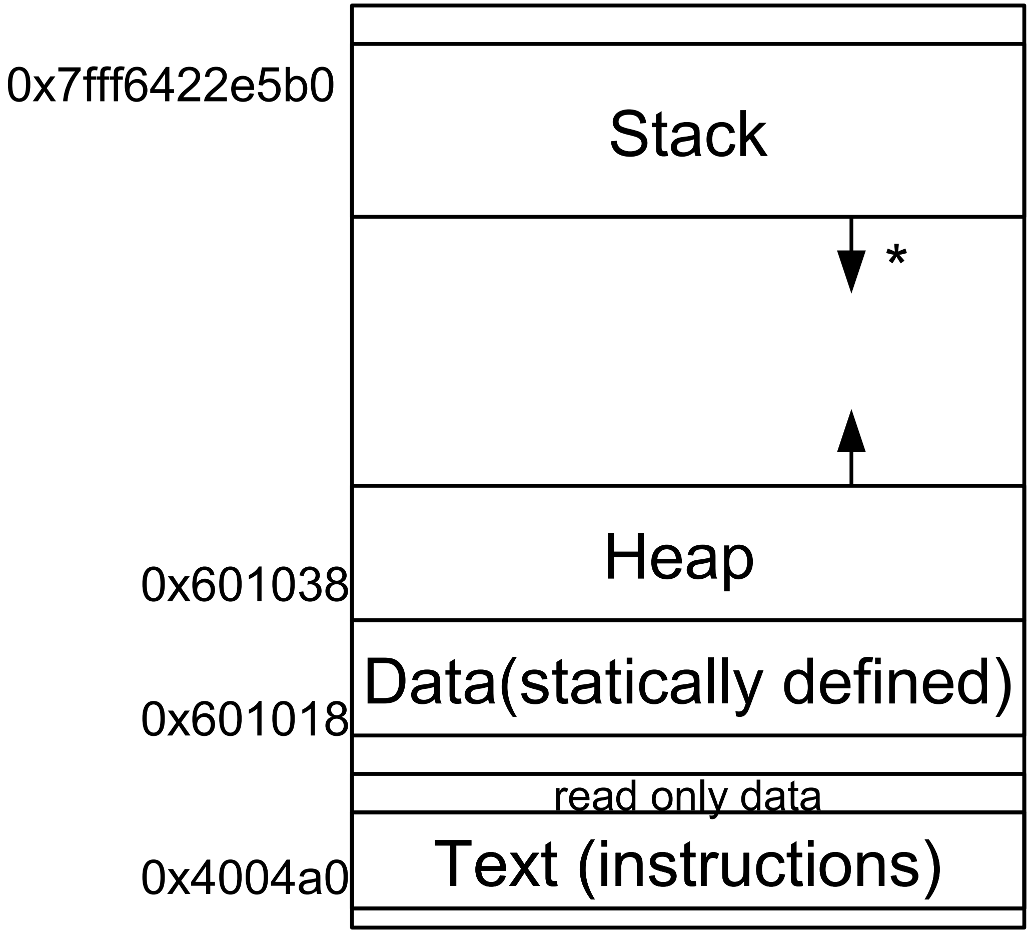
Memory Layout

readelf -a

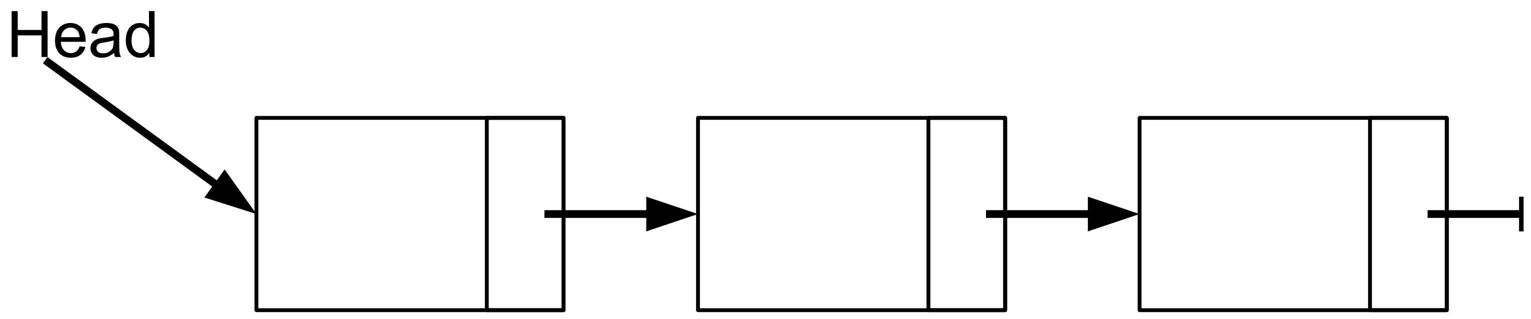
- What is in each section?

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

int gValue = 9;
int gArray[1024];

int main()
{
  int *pArray;
  int value = 10;
  printf("%d", gValue);
  pArray =        malloc(

  free(pArray);
  return 0;
}
```

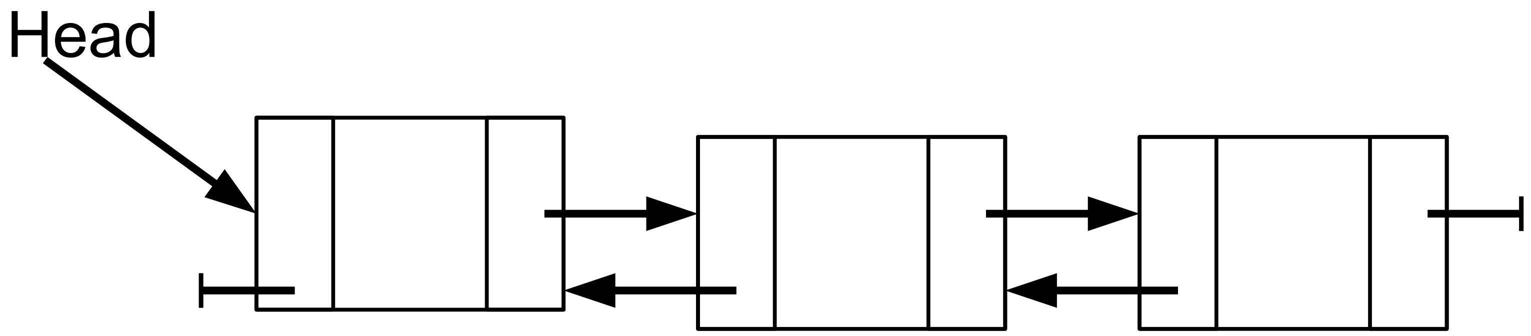
shared libraries?



Linked Lists



Doubly Linked Lists



Circular?

How to represent a node

```
struct Node
{
  int data;
  struct Node* psNext;
} Node;

Node sList;
Node *psList;
```

allocate?
deallocate?
access?

Which of these are legal?

```
sList.data = 5;
sList->psNext = NULL;
sList = NULL;
psList->data = 5;
psList = NULL;
```

Better C Definition for Node

```
typedef struct Node *NodePtr;

typedef struct Node
{
  int data;
  NodePtr psNext;
} Node;

Node sList;
NodePtr psList;
```

Problems

- Create an empty list pointed to by **psList**.
- Allocate space for a new node and set the list pointer to point to the new node.
- Place the integer **10** into the data field of the single node.
- Create another new node and place the integer **20** into the data field of the new node.
- Link the two nodes together placing the node with 20 after the node 10.
- A linked list exists pointed to by the list pointer **psList**. Write a function **length** that accepts the list pointer to a singly linked list and returns the length of the list.

Stack

```
typedef int DATATYPE;
```

```
typedef struct StackElement
{
  DATATYPE data; // the user data



} StackElement;

typedef  struct Stack
{



} Stack;
```

- stkCreate(
- stkDispose(
- stkPush(
- stkPop(
- stkPeek(

file input

previously used fgetc

```
#include <stdio.h>

int result, x, y;
FILE *pFile;
pFile = fopen(“data/test.txt”, “r”);

result = fscanf(pFile, “%d %d”, &x, &y);

fclose(pFile);

// what does fprintf() do?
```