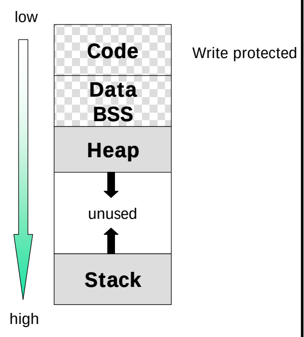
# malloc(): Dynamic Memory Management

Stanley Yao
Computer Science Department
University of Arizona

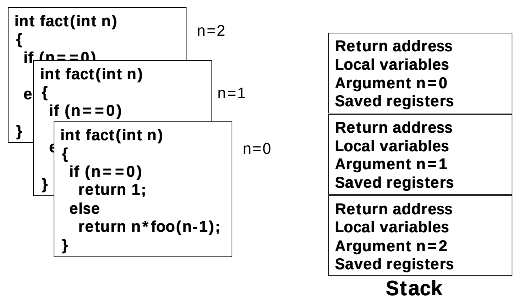
## Process Memory Layout

- Heap meets stack?
- Access the unused space?



## Function Calls & Stack

fact(2) = 2



```
int fact(int n)
{
  if (n==0)
e
}
```
n=2

```
int fact(int n)
{
  if (n==0)
}
```
n=1

```
int fact(int n)
{
  if (n==0)
    return 1;
  else
    return n*foo(n-1);
}
```
n=0

**Return address**
**Local variables**
**Argument n=0**
**Saved registers**

**Return address**
**Local variables**
**Argument n=1**
**Saved registers**

**Return address**
**Local variables**
**Argument n=2**
**Saved registers**

**Stack**

## malloc()

```
void *malloc(size_t n);
```

- Allocate n bytes of memory space in heap
- Sizeof() is recommended in expressing n
- Assign the starting address of this space to pointer p
- If no more space is available, return NULL
- The returned address from malloc() is void *
- It's suggested to cast the returned address to p's type

## Example 1

```
int *value;
value = (int *)malloc(sizeof(int));
If (value == NULL) {
  perror(“cs352”);
  exit(1);
}
*value = 10;
```

```
int *values;
values = (int *)malloc(3*sizeof(int));
If (value == NULL) {
          perror(“cs352”);
          exit(1);
}
values[0] = 11;
values[1] = 13;
values[2] = 14;
```

```
struct person *p;
p = (struct person *) malloc(sizeof(struct person));
If (value == NULL) {
          perror(“cs352 example”);
          exit(1);
}
p->age = 17;
```

## Example 2

```
int *value;
value = (int *) malloc(1);
*value = 10;
```

```
int *values;
values = (int *)malloc(2*sizeof(int));
values[0] = 11;
values[1] = 13;
values[2] = 14;
```

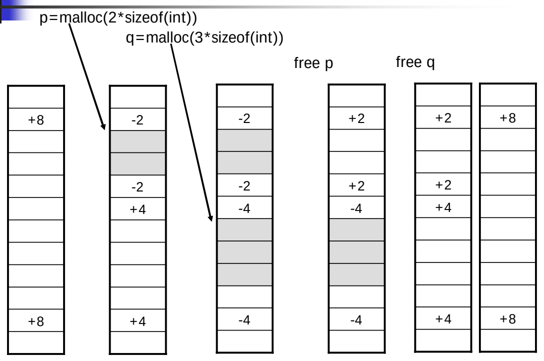
```
struct person *p;
p = (struct person *) malloc(sizeof(struct node));
p->age = 17;
```

## free()

```
void free(void *ptr);
```

- Return the previously allocated memory to the system
- In Java, GC will do this automatically. However in C, you must do it yourself ☹
- Never free a memory: memory leak
- Free a memory more than once: seg-fault

## How is Heap Organized?



## Electric Fence

- Stops your program on the exact instruction that overruns (or underruns) a malloc() memory buffer.
- Cooperate with gdb.
- Another tool: Dmalloc
  - Log
  - http://dmalloc.com/

## Thinking…

- How to detect a circular linking list?

## Acknowledgement

- John H. Hartman, *Classnotes for Csc352-Spring03*, CS Dept., University of Arizona, 2003
- Craig Chase, *Dynamic Memory Management (ppt)*, The University of Texas at Austin, 2002
- Brian W. Kernighan, Dennis M. Ritchie, *The C Programming Language (2nd Ed.),* Prentice Hall, 1988