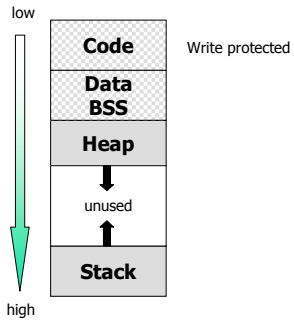
malloc(): Dynamic Memory Management

Alon Efrat
Computer Science Department
University of Arizona

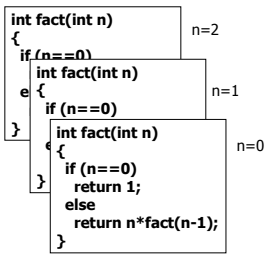
Process Memory Layout

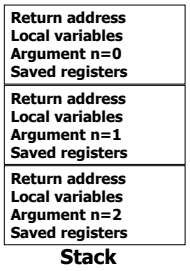
- Heap meets stack?
- Access the unused space?



Function Calls & Stack

fact(2) = 2





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;
```

5

Example 2

```
int *values;
values = (int *)malloc(3*sizeof(int));
values[0] = 11;
values[1] = 13;
values[2] = 14;
```

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

6

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

7

Example - Stack

```
// linkedList.cpp : Defines the entry point for the console application.
#include "stdafx.h"
#include<assert.h>

typedef struct node {
    int key ;
    struct node *next;
} NODE ;

//----------------------------------------
NODE * NewNode(){
    NODE *p = (NODE *) malloc( sizeof(NODE) ) ;
    assert( p != NULL ) ;
    p-> next = NULL ;
    p-> key = 0 ;
    return p ;
}
```

8

```
NODE * PopNode( NODE * phead ){
        assert( phead != NULL ) ;
        NODE * ptmp = phead -> next;
        free( phead ) ;
        return ptmp ;
}
```

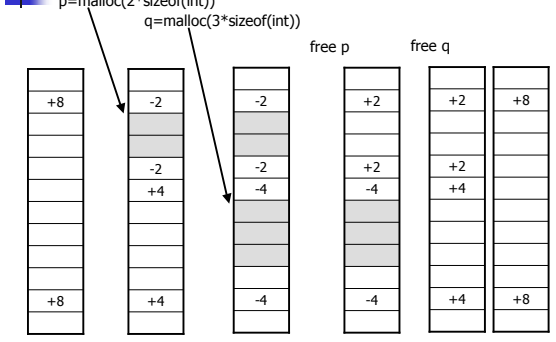
```
NODE *  PushNode( NODE *phead, int key
){
        NODE * pnew = NewNode();
        pnew->key = key ;
        pnew->next = phead ;
        return pnew ;
}
```

```
void PrintList( NODE * p ) {
    while( p != NULL ) {
      printf("Element: %d\n", p->key ) ;
      p=p->next ;
    }
}
```


Main()

```
NODE * pheader = NULL ;
int key ;  char c;
while(1){
        printf("Enter  a command for the Stack  (i/d/p/q) , followed by an int : " );
        while(  (c=getchar()) <= 'a' ||  c>'z' )
                ;
        if( c=='q')  exit(0);
        else if (c=='i') {
                scanf("%d", &key );
                pheader = PushNode( pheader, key );
        }
        else if(c=='d')  pheader = PopNode( pheader ) ;
        else if (c=='p') PrintList( pheader );
        else printf("Wrong command\n");
}
```


How is Heap Organized?

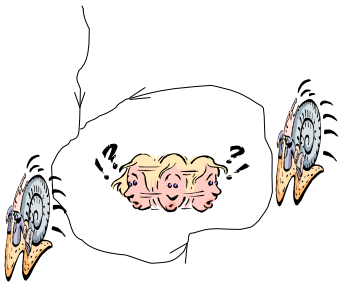


Electric Fence

- 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?



13

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

14