CSE216 Programming Abstractions

Dynamic Memory Allocation

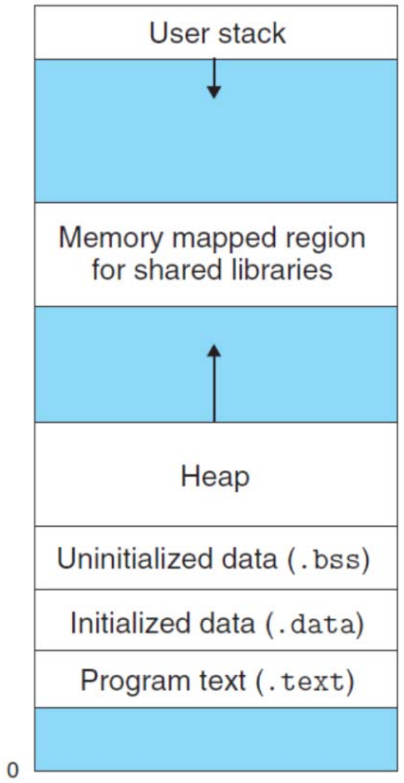
YoungMin Kwon

Dynamic Memory Allocation

- Allocate memory in heap

```
#include <stdlib.h>

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



Composite Types: struct

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

struct pair {
    char c;
    int  i;
};

int main() {
    struct pair s = { .c = 'a', .i = 99};
    struct pair *ps = NULL;
    printf("%c, %d, %p, %p, %p, %d\n", s.c, s.i, &s, &s.c, &s.i, sizeof(s));

    //allocate memory for ps
    ps = malloc(sizeof(struct pair));

    //use -> to reference a field of a pointer to a struct
    ps->c = 'a', ps->i = 99;
    printf("%c, %d, %p, %p, %p\n", ps->c, ps->i, ps, &(ps->c), &(ps->i));

    //deallocate memory for ps
    free(ps);
    return 0;
}
```

```
> a.exe
a, 99, 0061FEC4, 0061FEC4, 0061FEC8, 8
a, 99, 00991468, 00991468, 0099146C
```

Composite Types: union

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

union pair {
    char c;
    int  i;
};

int main() {
    union pair u = { .c = 'a', .i = 99};
    union pair *pu = NULL;
    printf("%c, %d, %p, %p, %p, %d\n", u.c, u.i, &u, &u.c, &u.i, sizeof(u));

    //allocate memory for pu
    pu = malloc(sizeof(union pair));

    //use -> to reference a field of a pointer to a union
    pu->c = 'a', pu->i = 99;
    printf("%c, %d, %p, %p, %p\n", pu->c, pu->i, pu, &(pu->c), &(pu->i));

    //deallocate memory for pu
    free(pu);
    return 0;
}
```

```
> a.exe
c, 99, 0061FEC8, 0061FEC8, 0061FEC8, 4
c, 99, 00681468, 00681468, 00681468
```

```c
//
// linkedlist.c
//
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct node {
    char *name;
    int id;
    //link to the next and the previous node
    struct node *next;
    struct node *prev;
};

struct node *make_node(char *name, int id) {
    //TODO: allocate memory of sizeof(struct node) bytes for n
    struct node *n =
    //TODO: allocate memory of strlen(name)+1 bytes for n->name
    n->name =
    //strcpy to n->name from name
    strcpy(n->name, name);
    //initialize other fields
    n->id = id;
    n->next = n->prev = NULL;
    return n;
}
```

```
void destroy_node(struct node *n) {
    printf("destroying node: %d, %s\n", n->id, n->name);
    //TODO: deallocate n->name
    //TODO: deallocate n
}

void add_node_between(struct node *pred,
                      struct node *succ,
                      struct node *n) {
    //TODO: insert n in between pred and succ
}

void remove_node(struct node *n) {
    //TODO: remove n from the linked list
}
```

```
struct node *get_list() {
    char name[256];
    int id;
    //make a head node (sentinel)
    struct node *head = make_node("head", -1);
    //circularly doubly linked list
    head->next = head->prev = head;

    while(1) {
        struct node *n;
        printf("enter name or quit to stop: ");
        scanf("%s", name);
        if(strcmp(name, "quit") == 0)
            break;
        printf("enter id: ");
        scanf("%d", &id);
        //TODO: make a node with name and id
        //TODO: add n to the last position of the list
    }
    return head;
}
```

```c
void print_list(struct node *head) {
    struct node *n;
    //TODO: from head->next until n reaches head, print n->id and n->name
    for( ; ; ) {
        printf("%3d: %s\n", n->id, n->name);
    }
}

void destroy_list(struct node *head) {
    while(head->next != head) {
        struct node *n = head->next;
        //TODO: remove n from the list
        //TODO: destroy n
    }
    //TODO: destroy head
}

int main() {
    struct node *head;
    head = get_list();
    print_list(head);
    destroy_list(head);
}
```