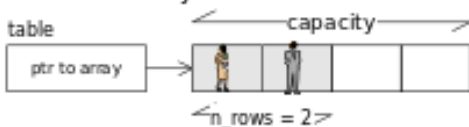


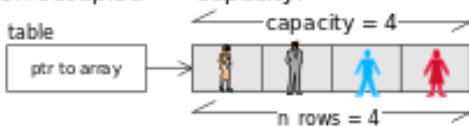
## Algorithm for Growable Array

**MAXIMUM  
OCCUPANCY  
4 PERSONS**

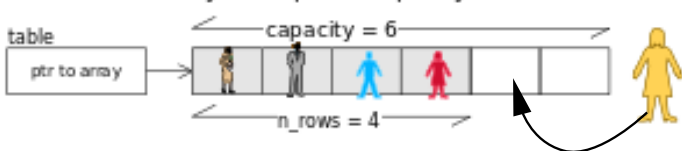
Add entries to array



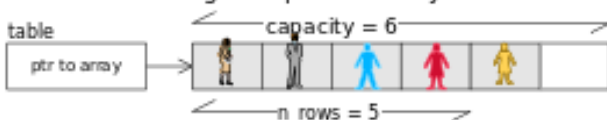
When occupied == capacity:



Then realloc memory and update capacity



Then continue adding to expanded array.

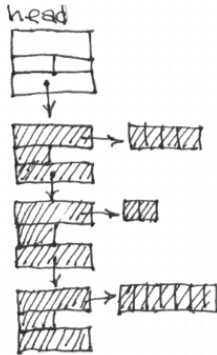


## 5 . A Linked List

### 5. ONE LINKED LIST

All structs and strings are dynamically allocated

linkp



Problem: realloc() may have to move the array

Solution: Create each entry as needed, insert in list

- 1 . use malloc to create each entry
- 2 . move pointers to insert entry in list

Defining the data structures:

```
struct link { char *word; int value; struct link *next; };  
struct link head;  
struct link *current_link;
```

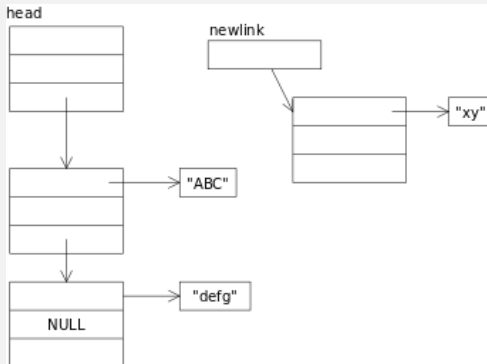
Examine Code: Discuss Algorithms for functions

Pros and Cons of this solution

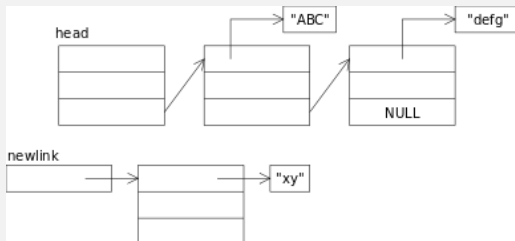
## Inserting a link at front of a linked list

To add a link to the front of a iinked list"

- 1 . make new link point to current first link
- 2 . make head link point to the new link

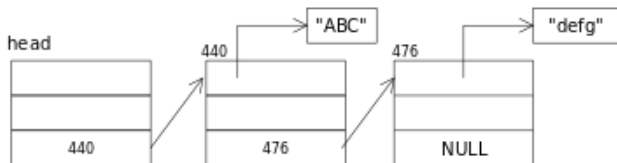


Linked lists are often drawn horizontally:



## Procedure for Inserting a New Item

**Start: List with two links**

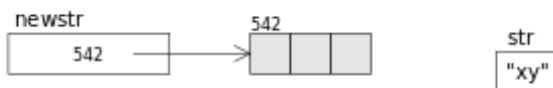


**Goal: add "xy" to list**

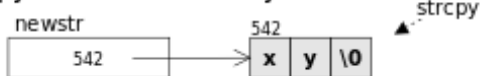
"xy"

## Make and populate new Link

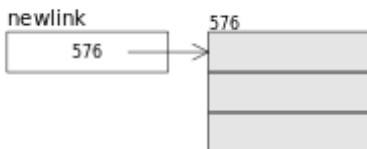
### Allocate new string array



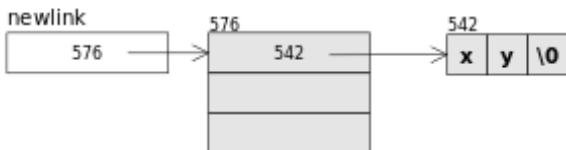
### Copy str to allocated array



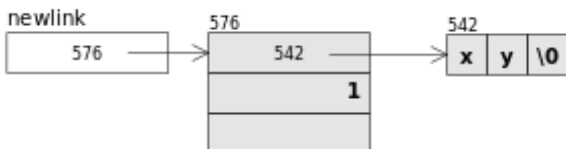
### Allocate new struct



### Store array pointer in struct

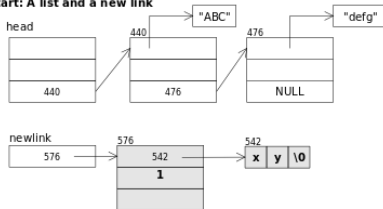


### Store value in struct

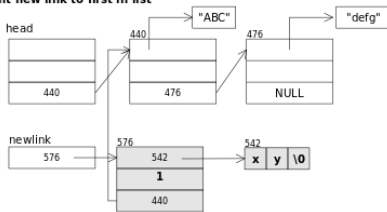


# Insert new Link at Front

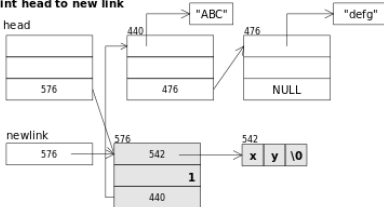
**Start: A list and a new link**



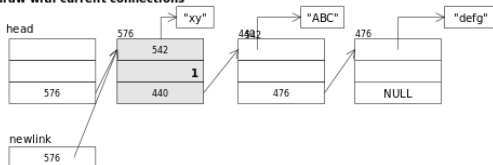
**Point new link to first in list**



**Point head to new link**

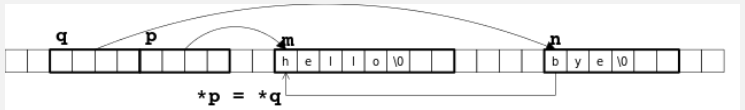


**Redraw with current connections**



## Common Use of Pointers: Processing Strings

```
char m[8] = "hello";  
char n[6] = "bye";  
strcpy(m, n);
```



You have to copy the string, char by char.

Use a loop with indexing `m[i] = n[i]` until `n[i] == \0`

or

Use a loop with pointers `*p = *q` until `*q == \0`