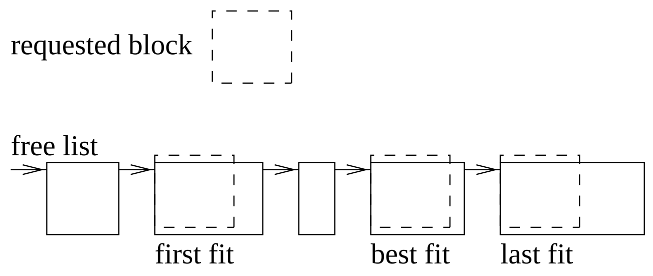
Memory allocation

- Allocation algorithms
- C functions
- Examples

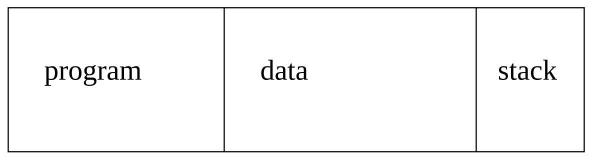
Allocation algorithms

- first fit – from the list of free blocks get first block bigger than requested one.
- last fit – from the list of free blocks get last block bigger then requested one.
- best fit – from the list of free blocks get the one minimizing the lost.

Allocation algorithms



Program memory



Memory allocation

```
void *malloc(size_t size);

void *calloc(size_t nelem, size_t elsize);

void *realloc(void *ptr, size_t size);

void free(void *ptr);
```

malloc – allocate requested block (return NULL if not available)

calloc – allocate space for array of `nelem` elements of `size` size.

realloc – change the size of allocated block (return NULL if change can not be done).

free – free allocated memory.

Example program - simple list

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

/*-------------------------------------------------*/
/* Node definition                                 */
/*-------------------------------------------------*/
struct node
{
struct node *next;
char *item;
};
```

```
/*-------------------------------------------------*/
/* Allocate new node structure and new item memory */
/*-------------------------------------------------*/
struct node *new_node( char *string )
{
struct node *tmp;

tmp = (struct node *) malloc( sizeof( struct node ) );
if (tmp == NULL)
  {
  printf( "No memory for node!\n\a" );
  exit( 1 );
  }
tmp->item = (char *) malloc( strlen( string )+1 );
if (tmp->item == NULL)
```

```
  {
  printf( "No memory for item!\n\a" );
  exit( 1 );
  }
strcpy( tmp->item, string );
tmp->next = NULL;
return tmp;
}

/*-------------------------------------------------*/
/* List of nodes                                   */
/*-------------------------------------------------*/
struct node *first_node = NULL;

/*-------------------------------------------------*/
```

```
/* Add new node to the list                             */
/*------------------------------------------------*/
void add_node( char *string )
{
struct node *tmp;

tmp         = new_node( string );
tmp->next   = first_node;
first_node  = tmp;
}
```

```
/*------------------------------------------------*/
/* Add all arguments to the list and print the list*/
/*------------------------------------------------*/
int main( int narg, char *argv[] )
{
int i;
struct node *ptr;

for (i=1; i<narg; i++)
  add_node( argv[i] );

ptr = first_node;
while (ptr != NULL)
  {
  printf( "Node: %s\n", ptr->item );
```

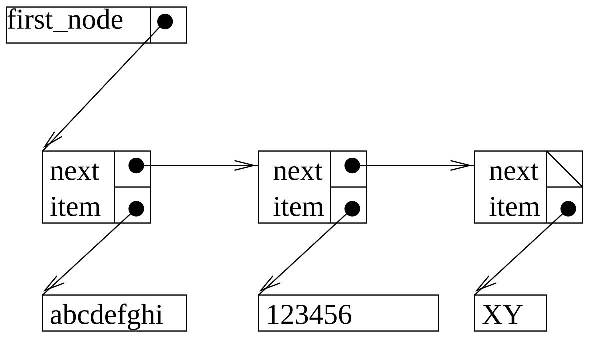
```
  ptr = ptr->next;
  }
}

/*------------------------------------------------*/
```

Example program - results

```
<jurek>(21)$./e1 XY 123456 abcdefghi
Node: abcdefghi
Node: 123456
Node: XY
```

Example program - data structure


first_node
next
item
next
item
next
item
abcdefghi
123456
XY