1

Outline

- In this lesson, we will:
  - Review that arrays can be great under certain circumstances
  - See that there are significant weaknesses
    - Fixed size
    - Expensive to move many entries
    - Same type
  - Understand that there must be other approaches

2

Arrays

- Up to now, if we want to store an arbitrary amount of data, we must use an array
  - We can create a structure with a fixed number of member variables, but we can't change that amount
- Recall that arrays occupy a block of memory
  - The memory is either allocated
    - Statically by the compiler on the stack, or
    - Dynamically by the operating system on the heap
  - In either case, you cannot go back and change the allocated memory

3

Arrays can be fast

- Arrays can be blindingly fast:
  - Accessing or changing an entry is very fast
    - Sometimes called *random access*
  - If it is sorted, we can search it quickly

4

Arrays

- Suppose you have an array with capacity 20, but it is currently storing 10 pieces of information:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
18	20	19	21	21	22	21	23	22	24										

- Next, suppose a new datum comes along that must be placed first, say 17:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
17	18	20	19	21	21	22	21	23	22	24									

  - This required us copy all ten entries over by one

The cost of memory

- Another issue is: suppose you normally only need to store on the order of 10 items, but in the worst case, you may have to store hundreds of items of data
  - Is it not a waste of memory to perpetually store a sufficiently large array that may be used a few minutes a day
- This is not an issue with larger general-purpose computing where memory is cheap, but consider an embedded system
  - Additional memory costs includes an increase in
    - Power consumption
    - Weight
    - Heat
    - Cost

The cost of memory

- Increasing any of these has cascading effects:
  - Increasing power consumption may either:
    - Decrease battery life
    - Increase the battery size
      - This further increases the weight
  - Increasing the weight may either:
    - Decrease portability
    - Increase weight-bearing structures
  - Increasing heat may either:
    - Require a larger heat sink and cooling apparatus
      - Thus further increasing the weight
    - Decrease lifetime
  - Increasing cost will decrease competitive advantages or decrease profits

Queues in operating systems

- This is also an issue with operating systems
  - There are many queues were, for example, an executing program is waiting
    - To use the printer
    - For additional memory
    - For input
- Most of these queues are empty or perhaps with one or two executing programs waiting some service
  - Unfortunately, the operating system must always be ready for an arbitrary number of executing programs waiting for some service…
  - Do we have an array of size 256 for each such service?
    - How about 512?

Concatenating arrays

- Suppose we have two arrays, and we want to concatenate them:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
27	39	82	84	57	82														

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
34	27	29	39	12	29	59	23												

  - Not a problem: copy the entries from the second to the first
  - Still slow if the second array is large

Concatenating arrays

- What do we do here?

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
27	39	82	84	57	82	58	48	27	37	85	94	23	75	23	85				

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
34	27	29	39	12	29	59	23	91	98	75	82	66	42						

  - What must we do now?
  - This will be even more expensive

Strings

- Suppose we have a string which we are editing:

  "A Elbereth Gilthoniel
  silivren penna míriel
  o menel aglar elenath!
  Na-chaered palan-díriel
  o galadhremmin ennorath,
  Fanuilos, le linnathon
  nef aear, sí nef aearon!"

- Strings are by default stored as arrays
  - How much work is required to edit an array?

J.R.R. Tolkien, "Lord of the Rings"

Text editors

- Similarly, in your IDE, you are editing your code:

```
#ifndef _LINUX_WAIT_H
#define _LINUX_WAIT_H


#include <linux/list.h>
#include <linux/stddef.h>
#include <linux/spinlock.h>
#include <asm/current.h>
#include <uapi/linux/wait.h>

typedef struct __wait_queue wait_queue_t;
typedef int (*wait_queue_func_t)(wait_queue_t *wait, unsigned mode, int flags, void *key);
int default_wake_function(wait_queue_t *wait, unsigned mode, int flags, void *key);

/* __wait_queue::flags */
#define WQ_FLAG_EXCLUSIVE       0x01
#define WQ_FLAG_WOKEN           0x02

struct __wait_queue {
        unsigned int flags;
        void *private;
        wait_queue_func_t func;
        struct list_head task_list;
};

/*
 * the following wait_bit_key_deprecated and wait_bit_queue_deprecated are
 * retained for 3rd-party modules that used PREPARE_WAIT_BIT before commit
 * "sched: Allow wait_on_bit_action() functions to support a timeout"
 */
struct wait_bit_key_deprecated {
        void *flags;
        int bit_nr;
};

struct wait_bit_queue_deprecated {
        struct wait_bit_key_deprecated key;
        wait_queue_t wait;
};

struct wait_bit_key {
        void *flags;
        int bit_nr;
#define WAIT_ATOMIC_T_BIT_NR -1
        unsigned long           timeout;
};

struct wait_bit_queue {
        struct wait_bit_key key;
        wait_queue_t wait;
};
```

As a file, it is an array of characters, but if you are editing it, you cannot store it as an array…

Summary

- Following this lesson, you now
  - Understand there are weaknesses in arrays
    - They cannot effectively be used in all situations
  - Know that the characteristics of arrays are sometimes weaknesses
  - Understand that we need a different approach to storing arbitrary amounts of information

13

References

[1] https://en.wikipedia.org/wiki/C++_classes
[2] https://en.wikipedia.org/wiki/Array_data_structure

14

Colophon

These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`.

The photographs of lilacs in bloom appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens on May 27, 2018 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.



15

Disclaimer

16