

W UNIVERSITY of WASHINGTON L05: Data Structures, Modules CSE333, Summer 2020

Using a Generic Linked List

- Type casting needed to deal with `void*` (raw address)
 - Before pushing, need to convert to `void*`
 - Convert back to data type when accessing

```
typedef struct node_st {
    void* element;
    struct node_st* next;
} Node;

Node* Push(Node* head, void* e); // assume last slide's code

int main(int argc, char** argv) {
    char* hello = "Hi there!";
    char* goodbye = "Bye bye.";
    Node* list = NULL;

    list = Push(list, (void*) hello);
    list = Push(list, (void*) goodbye);
    printf("payload: '%s'\n", (char*) ((list->next)->element));
    return 0;
}
```

manual_list_void.c 25

25

W UNIVERSITY of WASHINGTON L05: Data Structures, Modules CSE333, Summer 2020

C Header Files


- Header:** a file whose only purpose is to be `#include`'d
 - Generally has a filename `.h` extension
 - Holds the variables, types, and function prototype declarations that make up the interface to a module
- Main Idea:**
 - Every name `.c` is intended to be a module that has a name `.h`
 - name `.h` declares the interface to that module
 - Other modules can use name by `#include`-ing name `.h`
 - They should assume as little as possible about the implementation in name `.c`

30

30

W UNIVERSITY of WASHINGTON L05: Data Structures, Modules CSE333, Summer 2020

C Module Conventions (1 of 2)




- File contents:**
 - `.h` files only contain *declarations*, never *definitions*
 - `.c` files never contain prototype declarations for functions that are intended to be exported through the module interface
 - Public-facing functions are `ModuleName_functionname()` and take a pointer to "this" as their first argument
- Including:**
 - NEVER** `#include` a `.c` file – only `#include` `.h` files
 - `#include` all of headers you reference, even if another header (transitively) includes some of them
- Compiling:**
 - Any `.c` file with an associated `.h` file should be able to be compiled into a `.o` file
 - The `.c` file should `#include` the `.h` file; the compiler will check definitions and declarations for consistency

31

31

W UNIVERSITY of WASHINGTON L05: Data Structures, Modules CSE333, Summer 2020

C Module Conventions (2 of 2)



- Commenting:**
 - If a function is declared in a header file (`.h`) and defined in a C file (`.c`), *the header needs full documentation because it is the public specification*
 - Don't copy-paste the comment into the C file (don't want two copies that can get out of sync)
 - If prototype and implementation are in the same C file:
 - School of thought #1:** Full comment on the prototype at the top of the file, no comment (or "declared above") on code
 - School of thought #2:** Prototype is for the compiler and doesn't need comment; comment the code to keep them together

e.g. 333 project code

32

32

