ECS 36A, May 22, 2023

Announcements

1. Homework 3 and Extra Credit 3 are now due on May 30 (that’s the Tuesday after Memorial Day).
2. The gradescope for Extra Credit 3 is set up.
3. Homework 4 will be available on May 25 and will be due June 8 (last day of classes).
4. Final Exam is in this room, on June 9, from 10:30am–12:30pm
5. The web sites have been updated to May 19. If you don’t see something that should be there, please let me know.

Example: Dynamically Allocated Input Buffer

- Problem: *fgets* requires a maximum length to input
  - So it will fit into the input buffer without overflow
  - May read only part of a line
- Solution: write a function that will allocate space for any length line

Requirements

- Function must be able to input line of any length without knowing what that length may be
- Interface needs to be as similar to that of *fgets* as possible

Solution #1: For Interface

`char *dyngets(char *buf, int n, FILE *fp)`

- `char *buf`
  - If non-NULL, pointer to input buffer; *dyngets* acts exactly like *fgets*
  - If NULL, one line is stored in allocated space
- `int n`
  - size of array `buf`
  - *ignored* if `buf` is NULL
- `FILE *fp`
  - File pointer to source of input

Solution #2: Allocation

- Create a buffer that is preserved across calls
  - Use a static variable to point to this and the size of the buffer
- Static variable in function keeps variable and its value around after function returns

General Structure

- If buf is not NULL, call *fgets* and return its value
- Otherwise:
  1. Read a character; if end of file, go to step 6
  2. If there is room in the internal buffer, put character in and go to step 1
  3. If there is not room in the internal buffer, allocate (or reallocate) an internal buffer of length INCREMENT + length of current internal buffer
  4. Add the new INCREMENT to the length of the internal buffer
  5. Go to step 2
  6. Return pointer to internal buffer

Program Structure

- Main routine is *dyngets*
- It calls a function to insert the character
    - Allocation is done here

Main Routine

- Check to see if buf is non-NULL; if so, call *fgets* and return its return value
- Read characters, calling the insertion function for each
  - If EOF is read as the first character of the line, return NULL
  - Otherwise, tack on a newline if it is present
  - Terminate the internal buffer with '\0'
- Return pointer to internal space

Character Insertion Routine

- First, see if internal buffer is completely full
  - If so, increment the allocated space number
  - If nothing allocated yet, use malloc() to allocate the desired space
  - Otherwise, use realloc() to reallocate the space
- Append the character to the input line

Compiling With a Program

List multiple files for gcc

- For *dyngets*:

```
gcc –ansi –pedantic –Wall –g –o mcat mcat.c dyngets.c
```

- What is happening: for each file
  - Run the C preprocessor on the file to handle thee macros
  - Compile the file to produce an assembly language “.s” file
  - Assemble the resulting “.s” file to produce an object “.o” file
- Then for all files:
  - The linking loader merges all the “.o” files and some system libraries into an executable