CSE 333: Systems Programming

Section 4

Non-buffered IO and common bugs

Non-buffered IO

- So far we've mostly used *fopen*, *fread*, *fwrite*, and family
  - These return and use FILE* pointers and maintain per-file buffers
- For the current assignment and this section, we'll use *open*, *read*, *write*, and family
  - These return and use file descriptors (ints) and do not maintain buffers

Non-buffered IO

- Reasons to use non-buffered IO
  - Can implement different buffering/caching strategies on top of *read* and *write*
  - There is no equivalent of *fread* and *fwrite* for network and other IO devices aside from in third-party libraries
  - *open*, *read*, *write*, etc. translate directly to system calls, hence there is more explicit control

Non-buffered IO

- Syntactic differences
  - *read* and *write* take just the number of bytes as a parameter as opposed to *fread* and *fwrite*, which also take the number of elements
  - *open* takes flags and a mode as integers, as opposed to a mode string as with *fopen*
    - For example, use the flag *O_CREAT* to force file creation and the mode *S_IRUSR | S_IWUSR* to set the file permissions to 0600
    - See *man 2 open* for the full details

Common bugs

- Joe, Steve and I have seen some common bugs and related misconceptions throughout the course so far, so let’s go over some of them

Common bugs

- What is wrong with the following snippet of code? How do we fix it?

```
// Returns a copy of the middle
// third of the given string.
char* MiddleThird(const char* str) {
  size_t len = sizeof(str);
  char* copy;
  strncpy(copy, &str[len / 3], len / 3);
  return copy;
}
```

Common bugs

- What about in this scenario? Again, how can we fix the problem(s)?

```
typedef struct {
  int a, b;
} Payload;

void StorePayload(LinkedList list,
                  int a, int b) {
  Payload payload;
  payload.a = a;
  payload.b = b;
  Assert333(PushLinkedList(list, &payload));
}
```

Common bugs

- What about here?

```
// Payload is defined as before.
void RetrieveHeadPayload(
    LinkedList list, Payload payload) {
  if (NumElementsInLinkedList(list) == 0)
    return;
  LLIter iter = LLMakeIterator(list);
  Assert333(iter != NULL);
  LLIteratorGetPayload(iter, &payload);
}
```

Assignment for today

- Gain some experience with non-buffered IO
- Recognize and fix bugs related to improper use of string functions, stack-allocation versus heap-allocation, and incorrect error handling
- git pull to get the code; submit to the Dropbox
- Your solution should have no errors under Valgrind and should match the output of the sample solution, assuming mine is correct