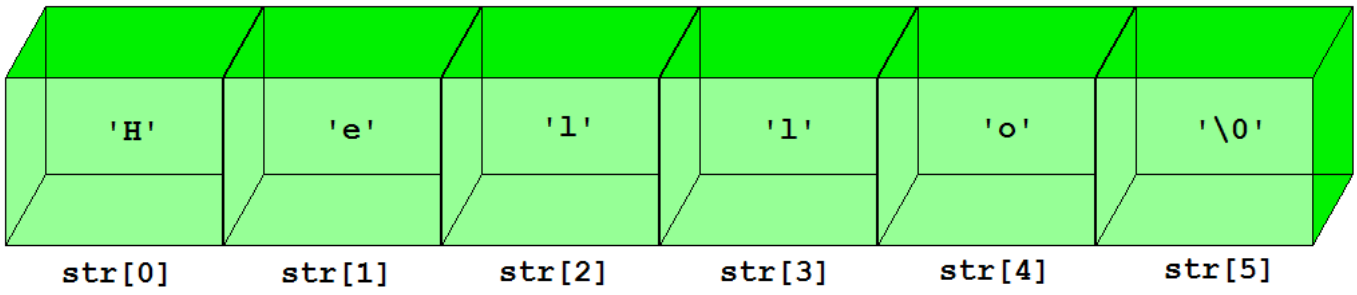
System File API

Buffering

- A buffer is simply a region of storage for bytes.
- Illustration of text in a buffer:



- Typically, a buffer is a contiguous region of memory, thereby facilitating the efficiency and accuracy of maintaining the buffer.
- Initially, a buffer contains invalid data. As a point of fact, all memory always represents either the value 1 or the value 0. Unless it is overwritten when allocated, a buffer, being a simple array of memory, holds any prior 1's and 0's. These residual values create the condition that the buffer is uninitialized and are sometimes referred to as "garbage" values.

- Once allocated, the buffer can be filled with valid data. Filling the buffer the first time is known as initialization.
- Depending on the amount of data available and the conditions for retrieving data, the buffer may or may not be filled completely. It is for this reason that the last valid byte must be tracked.
- When bytes are requested from the buffer, the index of the start byte is simply moved to the next position beyond the last byte read.
- Generally then, additional reads from the buffer simply move the index of the start byte again.
- Each time a request to read bytes from the buffer occurs however, the buffer must be checked for remaining valid content. When insufficient data remains, the buffer must be refilled. After refilling, the index of the start byte and the index of the last byte must be reset.
- Depending on how the buffer is used, any remaining data may need to be shifted within the buffer prior to refilling.

- Pseudocode examples:

```
// Rudimentary version – buffer is completely
//  overwritten
// Global variables:
//    buffer
//    start_byte_index
//    last_byte_index
//    no_more_data (boolean); initialized to False
refill_buffer()
    number_of_bytes_read= read( file, buffer,
     buffer_size )
    if number_of_bytes_read > 0
        start_byte_index= 0
        last_byte_index= number_of_bytes_read - 1
    else
        no_more_data= True
```

```
// Alternate version – bytes are permitted to remain
//  in buffer during refill operation
refill_buffer()
    if start_byte_index <= last_byte_index
        number_of_remaining_bytes= last_byte_index -
         start_byte_index + 1
        shift remaining bytes to beginning of buffer
    else
        number_of_remaining_bytes= 0
    number_of_bytes_read= read( file, buffer +
     number_of_remaining_bytes, buffer_size –
     number_of_remaining_bytes )
    bytes_in_buffer= number_of_remaining_bytes +
      number_of_bytes_read
    if bytes_in_buffer > 0
        start_byte_index= 0
        last_byte_index= bytes_in_buffer - 1
    else
        no_more_data= True
```

```
// Read from buffer
read_from_buffer( destination, count )
    if no_more_data
        return 0
    if refill_is_needed( count )
        refill_buffer()
        if no_more_data
            return 0
    bytes_in_buffer= last_byte_index – start_byte_index
     + 1
    bytes_to_read= min( count, bytes_in_buffer )
    copy bytes_to_read bytes from buffer into
     destination
    start_byte_index= start_byte_index + bytes_to_read
    return bytes_to_read //… which is the number read

// Determine whether buffer needs refilling
// Q: Can the current contents of the buffer meet our
//  current need?
refill_is_needed( count )
    if ( last_byte_index - start_byte_index + 1 ) >=
     count
        return False // no refill needed
    return True // refill needed
```