Clase #12 de 29
Flujos de Texto & Máquinas de Estado

Jun 24, Martes
Jun 25, Miércoles

Introducción a Flujos de Texto & Líneas

K&R 1.5.3 Conteo de Líneas

Flujos de Texto

- Definición de Flujo de texto
  - Secuencia de líneas
    - Secuencia de caracteres finalizada por el carácter nueva línea ('\n')
- Flujos de texto estándar
  - Entrada: stdin
  - Salida: stdout
  - Error: stderr
- Redirección
  - hola.exe > salida.txt
  - ordenar.exe < in.txt > out.txt

- La definición del **modelo** de flujo de texto es **única** y conocida
- Pero **cada entorno** de ejecución tiene su **propia representación**
- La **biblioteca abstrae** los detalles de implementación y **presenta al programador** los flujos de texto de una **única forma**, la del modelo.

Abstracción por medio del modelo

- Diferentes representaciones de líneas de texto
  - Mac OS 9
    - CR 13
  - Windows
    - CR+LF 13 10
  - Unix y derivados (Mac OS X)
    - LF 10
  - Muchas representaciones más, por ejemplo
    - Longitud
    - Fija con Espacios
- Modelo
  - Abstracción de la representación
  - Responsabilidad.

Ejemplo de aplicación de la abstracción

- Origen
  - Conceptual
    - ABC
    - DE
  - Modelo
    - A B C \n D E \n
    - 65 66 67 10 68 69 10
- Lectura

```
while( (c=getchar()) != EOF )
  printf("%d ", c);
```

- Diferentes implementaciones
  - Mac Os 9
    - Origen 65 66 67 13 68 69 13
    - Salida **65 66 67 10 68 69 10**
  - Windows
    - Origen 65 66 67 13 10 68 69 13 10
    - Salida **65 66 67 10 68 69 10**
  - Unix, Mac OS X
    - Origen 65 66 67 10 68 69 10
    - Salida **65 66 67 10 68 69 10**
  - Hypothetical OS (Longitud)
    - Origen 3 65 66 67 2 68 69
    - Salida **65 66 67 10 68 69 10**

En Resuemn

- En la práctica
  - En Posix no hay diferencia entre modo texto y binario
  - En Windows
    - Modo binario no tiene conversiones
    - Sí Hay conversiones para texto
      - 10 (\n)
        - \n se ecribe como 13 10
        - 13 y 10 se leen como \n
      - 26 (SUB)
        - Se escribe igual
        - Cuando se leen se interpreta que después no hay más datos, aunque no se el caso

FILE Struct en Diferentes Implementaciones

```
// Unix like
typedef struct __sFILE {
   unsigned char *_p; /* current position in (some) buffer */
   int _r; /* read space left for getc() */
   int _w; /* write space left for putc() */
   short _flags; /* flags, below; this FILE is free if 0 */
   short _file; /* fileno, if Unix descriptor, else -1 */
   struct __sbuf _bf; /* the buffer (at least 1 byte, if !NULL) */
   int _lbfsize; /* 0 or -_bf._size, for inline putc */

   /* operations */
   void *_cookie; /* cookie passed to io functions */
   int (* _Nullable _close)(void *);
   int (* _Nullable _read) (void *, char *, int);
   fpos_t (* _Nullable _seek) (void *, fpos_t, int);
   int (* _Nullable _write)(void *, const char *, int);

   /* separate buffer for long sequences of ungetc() */
   struct __sbuf _ub; /* ungetc buffer */
   struct __sFILEX *_extra; /* additions to FILE to not break ABI */
   int _ur; /* saved _r when _r is counting ungetc data */

   /* tricks to meet minimum requirements even when malloc() fails */
   unsigned char _ubuf[3]; /* guarantee an ungetc() buffer */
   unsigned char _nbuf[1]; /* guarantee a getc() buffer */

   /* separate buffer for fgetln() when line crosses buffer boundary */
   struct __sbuf _lb; /* buffer for fgetln() */

   /* Unix stdio files get aligned to block boundaries on fseek() */
   int _blksize; /* stat.st_blksize (may be != _bf._size) */
   fpos_t _offset; /* current lseek offset (see WARNING) */
} FILE;
```

```
// Microsoft
struct _iobuf {
   char* _ptr;
   int _cnt;
   char* _base;
   int _flag;
   int _file;
   int _charbuf;
   int _bufsiz;
   char* _tmpfname;
};
typedef struct _iobuf FILE;
```

Experimentación con Flujos de Texto & Líneas

K&R 1.5.3 Conteo de Líneas

Ejemplos: Cat y Hexdump

A continuación, se muestra una secuencia de comandos que presentan el contiendo de archivos de dos formas

- Comando **cat:** como texto
  - Se muestra el carácter asociado a cada byte
  - Recordar que hay caracteres que son de control, como \n
  - En Windows el comando **type** es similar
- Comando **hexdump:** como números
  - Se muestra el valor de cada byte, en base hexadecimal
  - También se muestran el offset y los caracteres que no son de control
  - Los caracteres de control se muestran con un punto (.)
  - En Windows el comando **debug** es similar

Archivos ejemplos

- **TestA.txt**
  - solo tiene el carácter **A**, es decir el byte **65** en base decimal o el **41** en base hexadecimal
- **TestLF.txt**
  - Dos líneas, ABC y DE, en formato Unix, es decir byte **\n** como terminador de línea
- **TestCR.txt**
  - Ídem pero para MacOS classic, es decir, **\r** como terminador
- **TestCRLF.txt**
  - Ídem pero para Windows, es decir, **\r\n** como terminador

```
$ Azul: Símbolo de commandos y commando
Negro: Salida del comando

$
A$
00000000  41                                        |A|
00000001

$
ABC
DE
$
00000000  41 42 43 0a 44 45 0a                     |ABC.DE.|
00000007

$
$
00000000  41 42 43 0d 44 45 0d                     |ABC.DE.|
00000007

$
ABC
DE
$
00000000  41 42 43 0d 0a 44 45 0d 0a
|ABC..DE..|
00000009

$
```

- ¿Por qué cat TestCR.txt *parace* no mostrar información?
- ¿Qué ocurre si se lee cada uno de estos cuatro archivos con un flujo en modo texto en Mac OS 9, Windows, y Unix?

Diferencia entre Flujos de Texto y Flujos Binarios

Escritura por Flujo de Texto

- Texto
  - ABC
  - DE
- Representación según modelo
  - A B C \n D E \n
- Destino: Archivo "Test.ssl"
- *Ejercicio*
  - Escribir un programa que imprima las dos líneas
  - Determinar la representación en bytes del archivo para cada entorno
    - Entorno Windows
    - Entorno Mac OS 9
    - Entorno Hypothetic OS
      - Longitud representada con unsigned int de 32 bits
    - Entorno Fixed Maximum Width
      - Determinar delimitador, tamaño y relleno

Leer desde de Flujo de Texto o de Flujo Binario

- Código fuente
  - ¿Cómo mostrar el entero asociado a cada carácter?
- Origen: Archivo "Test.ssl"
- Lectura a través de flujo de texto
- Lectura a través de flujo binario
- Conclusión
  - ¿Cuál es la diferencia?
    - Conversión
- *Ejercicio*
  - Escribir programa para mostrar valor de cada byte
  - Determinar salida de ese programa para cada entorno, para flujo de texto y para flujo binario
    - Entorno Windows
    - Entorno Mac OS 9
    - Entorno Hypothetical OS
    - Entorno Fixed Maximum Width

Demostración con ContarLineas

```
#include <stdio.h>

// count lines in input
int main(){
  int nl=0;

  for (int c; (c = getchar()) != EOF;)
    if (c == '\n')
      ++nl;
  printf("%d\n", nl);
}
```

- Literal Carácter o Constante Carácter
- Valor de un Literal Carácter
- Tipo de un Literal Carácter
- Buffer
  - Límite
  - Comienzo del procesamiento
- Señal de fin de ingreso
- Redirección
  - >
  - <
  - Pipes |
- Ejemplos
  1. `cat lc.c`
  2. `make lc`
  3. `find . -name "lc*"`
  4. `ls lc.*`
  5. `./lc`
  6. `./lc < lc.c`
  7. `./lc < lc.c > out.txt`
  8. `cat out.txt`
  9. `./lc < lc.c | ./lc`

Introducción a Máquinas de Estado

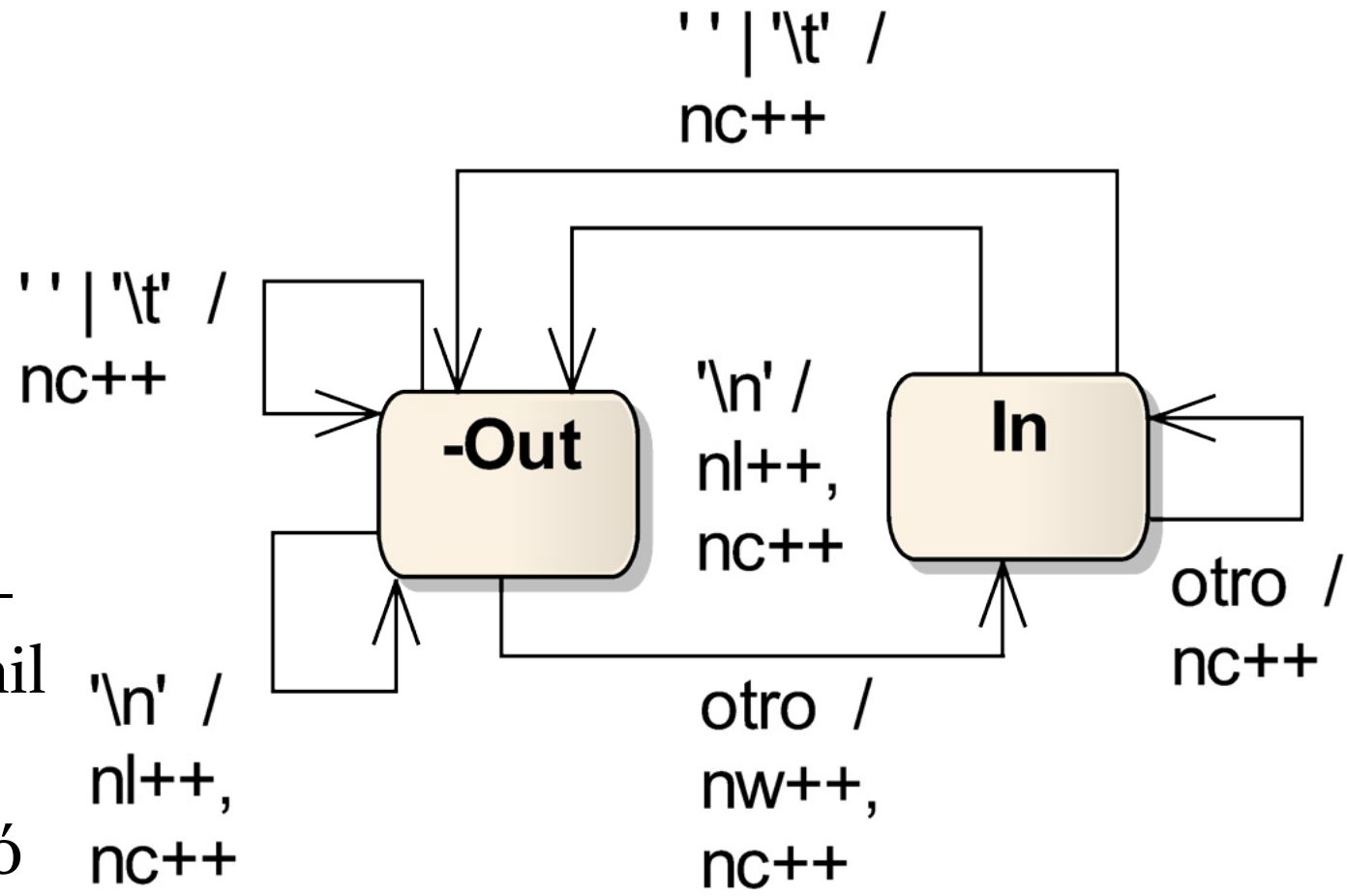
K&R 1.5.4 Conteo de Palabras

Problema

- Contar palabras, líneas y caracteres
- ¿Qué es una palabra?
- Diseño lógico
- Tecnología de implementación
- Abstracciones disponibles
  - Paradigma tipo imperativo
    - Paradigma procedural
      - Estilo estructurado
      - Estilo lineal, sin estructura
  - Paradigma tipo declarativo

Solución

- Máquina de estados con acciones
  - Diagrama de transiciones – Notación Símil UML
- Psuedocódigo ó
- Algoritmo.



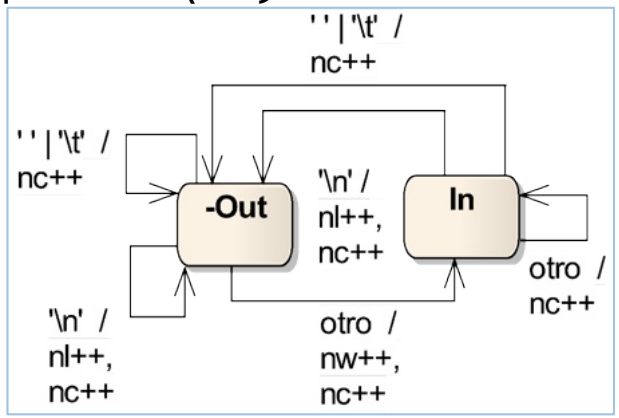
Implementación y Generalización

```c
#include <stdio.h>
#define IN  1 /* inside a word  */
#define OUT 0 /* outside a word */

int main(){
  int c, nl, nw, nc, state;
  state = OUT;   nl = nw = nc = 0;

  while ((c = getchar()) != EOF){
    ++nc;
    if (c == '\n')
      ++nl;
    if (c==' ' || c=='\n' || c=='\t')
      state = OUT;
    else if (state == OUT){
      state = IN;
      ++nw;
    }
  }

  printf("%d %d %d\n", nl, nw, nc);
  return 0;
}
```




```c
#include <stdio.h>
#define IN  1 /* inside a word  */
#define OUT 0 /* outside a word */

int main(void){
   int c, nl, nw, nc, state;
   state = OUT;   nl = nw = nc = 0;

   while ((c = getchar()) != EOF)
      if (state == OUT )
         if (c == '\n'){
            ++nl, ++nc;
            state = OUT;
         }
         else if (c==' ' || c=='\t'){
            ++nc;
            state = OUT;
         }
         else {
            ++nw, ++nc;
            state = IN;
         }
      else /* IN */
         if (c == '\n'){
            ++nl, ++nc;
            state = OUT;
         }
         else if (c==' ' || c=='\t'){
            ++nc;
            state = OUT;
         }
         else {
            ++nc;
            state = IN;
         }

   printf("%d %d %d\n", nl, nw, nc);
   return 0;
}
```

Ejercicio

- Implementar el programa de la sección 1.5.4 para que utilice **enum** en vez de **define** y **switch** en vez de **if**.

Términos de la clase #08

Definir cada término con la bibliografía

- Experimentación con Flujos de Texto y Líneas
  - Comandos cat y hexdump de Unix ó comandos type y debug de Windows
  - Diferencia entre Flujo Binario y Flujo de Texto
  - Flujo binario
  - Encadenamiento o entubamiento (Pipe) de comandos, la salida de un proceso a la entra de otro
- make sin makefile
- Carácter literal o constante carácter
- Valor y tipo de un literal carácter
- Entrada con buffer

Tareas para la próxima clase

1. Leer Entrada-Salida de a Caracteres y Redirección https://josemariasola.wordpress.com/papers#CharacterInputOutputRedirection

¿Consultas?

Fin de la clase