§8.3-8.6: Low Level I/O

•*devo*

31 Oct 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

- ***journals** in folder*

TRINITY WESTERN UNIVERSITY

http://cmpt14x.seanho.com/

Review of last time (8.1-8.2)

- Number bases:
  - Binary
  - Hexadecimal (0BEEFH)
  - Octal (115B)
    - Defining characters with octal: 115C
- Units of measure of memory:
  - Bits, nibbles, bytes, words, pages
- Units of measure for hard disks:
  - C/H/S geometry
- SI units vs binary units, KB vs. Kb, etc.

What's on for today (8.3-8.6)

- SYSTEM module
  - LOC, ADDRESS, ADR, CAST (vs. VAL)
  - M2 variables pointing to specific memory
- Files:
  - Logical/program/physical files
  - text/binary streams, channels
- Sequential streams: StreamFile, *IO libraries
- Rewindable streams: SeqFile, *IO libraries
  - Reread and Rewrite
  - File modes: read/write/old

The SYSTEM module

- M2 allows us some direct low-level access:
  - Lots of goodies can be imported from the SYSTEM library
  - It's not really a library (no DEF/IMP), but a pseudo-module provided by the compiler
  - Importing from SYSTEM flags our program as non-portable: it uses low-level implementation-specific features
- SYSTEM is a power tool:
  - You can shoot yourself in the foot pretty badly!

Features of SYSTEM

- Some things we can import from SYSTEM:
  - TYPE LOC;
    - Data storage units have this type
  - TYPE ADDRESS;
    - Storage locations (memory address) have this type
  - PROCEDURE ADR (VAR v): ADDRESS;
    - Returns the storage address of a variable
  - PROCEDURE CAST (<type>; val): <type>;
    - Similar to VAL, but unsafe conversion

CAST vs. VAL

- Both CAST and VAL convert types:

```
VAR myCard : CARDINAL; myReal : REAL;
myCard := VAL (CARDINAL, myReal);
myCard := CAST (CARDINAL, myReal);
```

- But VAL converts the value of myReal to a meaningfully equivalent CARDINAL value
  - (e.g., truncates the real number)
  - Safe type conversion
- CAST re-interprets myReal's bit pattern as a CARD
  - Re-interpretation could lead to weird value
  - Unsafe type conversion

Accessing specific memory

- You can declare a variable to refer to a particular location in the computer's memory:

```
VAR keyboard [0C000H] : CHAR;
```

- This example points to one byte (CHAR) at location $C000_{16}$ in memory
  - This could be part of the keyboard buffer
  - Or a pixel on the screen
  - Or memory used by other programs
  - Very dangerous!
- Memory protection: OS prevents one program from accessing another program's memory

Example: Generic swap

```
MODULE Swaps;
FROM SYSTEM IMPORT LOC;

PROCEDURE CanSwap (a, b: ARRAY OF LOC) : BOOLEAN
BEGIN
    RETURN HIGH (a) = HIGH (b);
END CanSwap;

PROCEDURE Swap (VAR a, b : ARRAY OF LOC);       (* any type! *)
VAR
    temp : LOC;
    max, count : CARDINAL;
```

Generic swap, cont.

```
BEGIN
    IF CanSwap (a, b)
        THEN
            FOR count := 0 TO HIGH (a)    (* swap one LOC at a time *)
                DO
                    temp := a [count];
                    a [count] := b [count];
                    b [count] := temp;
                END;
        END;
END Swap;


END Swaps.
```

Files

- A logical file is an abstract concept in the programmer's mind
- A program file is generally a variable in the code that refers to a file
- A physical file is a recording of a logical file – e.g., in magnetic media on a hard disk
- A stream is a sequence of data items of the same type, from an origin to a destination
  - Text stream: items regarded as characters
  - Binary stream: items regarded as bits
- A stream flows through a channel

Streams and files

- STextIO, SRealIO, etc. work on text streams
- Sequential files are organized as streams:
  - Can be written only at the end (appended)
    - e.g., output stream to speakers
  - Two kinds of sequential text files:
    - Restricted stream:
      - **Read from start; write to current position**
    - Rewindable sequential stream:
      - **Can also rewind current position back to start**
- Random-access files can be indexed:
  - Write at a given position within the file

Restricted stream I/O

- The StreamFile library opens and closes sequential files
- The TextIO, RealIO, etc. libraries contain the same procedures as their S-equivalents, but
  - Each procedure has an extra param specifying the file channel to use:

```
VAR
    out: ChanId;                    (* file channel *)
    result : OpenResults;
StreamFile.Open (out, "output.txt", write, result);
WholeIO.WriteCard (out, myCard, 0);
StreamFile.Close (out);
```

Standard input, standard output

- The standard input and output channels (usually keyboard and screen) are file channels:

```
WholeIO.ReadCard (StdChans.StdInChan(), myCard);
WholeIO.WriteCard (StdChans.StdOutChan(), myCard,
  0);
```

- With StreamFile, you can have multiple channels open at the same time
  - e.g., output to screen and file simultaneously
  - Now we can understand how RedirStdIO works

Rewindable sequential stream I/O

- The SeqFile library opens and closes rewindable sequential streams:
  - OpenRead, OpenWrite, OpenAppend
  - Reread (cid: ChanId): rewind to beginning
  - Rewrite (cid: ChanId): clear file and start over
- Open streams with a combination of modes:
  - Read, write, old
    - Old: ok to overwrite (clobber) existing files
- If opened read+write, use Reread/Rewrite to switch between reading and writing

Example: rewindable file

- Read from keyboard, store in file, read back:

```
FROM SeqFile IMPORT
    ChanId, OpenWrite, write, read, old, Close, Reread,
      OpenResults;
FROM WholeIO IMPORT
    ReadCard, WriteCard;
FROM StdChans IMPORT
    StdInChan, StdOutChan;
VAR
    file : ChanId;
    result : OpenResults;
BEGIN
    OpenWrite (file, “output.txt”, read+write+old, result);
```

Example: rewindable file, cont.

```
IF result = opened
    THEN
        WriteString (StdOutChan(), “Type a number: ”);
        WriteLn (StdOutChan());
        ReadCard (StdOutChan(), myCard);
        WriteCard (file, myCard, 0);

        Reread (file);          (* rewind file and start reading *)
        ReadCard (file, myCard);
    END;
Close (file);
```

Review of today (8.3-8.6)

- SYSTEM module
  - LOC, ADDRESS, ADR, CAST (vs. VAL)
  - M2 variables pointing to specific memory
- Files:
  - Logical/program/physical files
  - text/binary streams, channels
- Sequential streams: StreamFile, *IO libraries
- Rewindable streams: SeqFile, *IO libraries
  - Reread and Rewrite
  - File modes: read/write/old

TODO items

- No lab this week!
- Homework due Wed: 8.13 #44
- Quiz ch8 on Fri
- Reading: through §9.6 for Wed