Class Information

ANNOUNCEMENTS

MIDTERM EXAM:

• All homework solutions have been posted.
• Exam covers all 4 homeworks, all lectures until dynamic vs. lexical scoping (basic idea), compiler part of project 1.
• Please come 5 - 10 minutes early so we can start on time.
  – Leave backpacks and jackets in front of the room.
  – No electronics (e.g.: smart phones)
  – Don’t forget to bring your Rutgers ID. If we don’t know who you are, you cannot take the exam.
  – No bathroom break. Sorry.
Stack Frame, Activation Record

Scott: Chap. 8.1 - 8.2; ALSU Chap. 7.1 - 7.3

- Run-time stack contains frames for main program and each active procedure.

- Each stack frame includes:
  1. Pointer to stack frame of caller (**control link** for stack maintainance and dynamic scoping)
  2. Return address (within calling procedure)
  3. Mechanism to find non-local variables (**access link** for lexical scoping)
  4. Storage for parameters, local variables, and final values

```
Frame Pointer (FP)

parameters
return value
return address
access link
caller FP
locals
```

fall’17, lecture 15 page 2
Context of Procedures

Two contexts:

- **static** placement in source code (same for each invocation)
- **dynamic** run-time stack context (different for each invocation)

Scope Rules

Each variable reference must be associated with a single declaration (i.e., an offset within a stack frame).

Two choices:

1. Use static and dynamic context: *lexical scope*
2. Use dynamic context: *dynamic scope*

- Easy for variables declared locally, and same for *lexical* and *dynamic* scoping
- Harder for variables not declared locally, and not same for *lexical* and *dynamic* scoping
Lexical Scoping Example

**scope of a declaration**: Portion of program to which the declaration applies

Program

```
x, y: integer  // declarations of x and y
begin
  Procedure B  // declaration of B
    y, z: real  // declaration of y and z
    begin
      ...
      y = x + z  // occurrences of y, x, and z
      if (...) call B  // occurrence of B
    end
  Procedure C  // declaration of C
    x: real  // declaration of x
    begin
      ...
      call B  // occurrence of B
    end
  ...
  call C  // occurrence of C
  call B  // occurrence of B
end
```
Lexical Scoping Example

Calling chain: MAIN ⇒ C ⇒ B ⇒ B
Scoping and the Run-time Stack

Access links and control links may be used to look for non-local variable references.

**Static Scope:**

Access link points to stack frame of the most recently activated lexically enclosing procedure

⇒ Non-local name binding is determined at compile time, and implemented at run-time

**Dynamic Scope:**

Control link points to stack frame of caller

⇒ Non-local name binding is determined and implemented at run-time
Next Lectures Roadmap

- More on non-local data access (access links and displays)
- Parameter passing styles: Read Scott Chapter 8.3
- Introduction to functional languages; read Scott Chapter 10
- Lambda calculus