

CSc 520

Principles of Programming Languages

22 : Names, Scope, Bindings — Nested Procedures

Christian Collberg

collberg+520@gmail.com

Department of Computer Science
University of Arizona

—Spring 2008 — 22

Copyright © 2008 Christian Collberg

[1]

Nested Subroutines

- Algol 60, Pascal, Ada, Modula-2, etc. allow procedures to be nested inside each other.
- **Closest nested scope rule:**
 - a name that is introduced in a declaration is known in the scope in which it is declared, and in each internally nested scope, unless it is hidden by another declaration of the same name.
 - To search for the declaration corresponding to a use of a name, we search outward from the current scope.
- Nested subroutines are able to access the parameters and local variables of surrounding scopes.

520 —Spring 2008 — 22

[2]

Nested Subroutines...

```
procedure P1 (A1:T1);
var X : real;
  procedure P2 (A2: T3);
    procedure P3 (A3 : T3);
      begin (* body of P3 *) end;
  begin
    (* body of P2 *)
  end;
  procedure P4 (A4: T4);
    function F1 (A5 : T5);
      var X : integer;
      begin (* body of F1 *) end;
  begin
    (* body of P4 *)
  end;
begin
  (* body of P1 *)
end;
```

—Spring 2008 — 22

Accessing Non-Local Variables

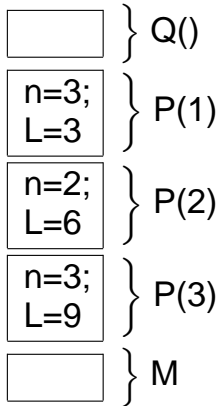
```
PROGRAM M;
  PROC P(n);
    LOCAL L;
    PROC Q(); BEGIN PRINT L; END Q;
  BEGIN
    L := n * 3;
    IF n >= 1 THEN P(n-1) ELSE Q() ENDIF;
  END P;
```

- Which L should Q print? There are three Ls on the stack to choose from!

520 —Spring 2008 — 22

Accessing Non-Local Variables

```
PROGRAM M;
  PROC P(n);
    LOCAL L;
    PROC Q();
      BEGIN PRINT L; END Q;
  BEGIN
    L := n * 3;
    IF n >= 1
      THEN P(n-1);
      ELSE Q();
    ENDIF;
  END P;
BEGIN P(3); END M.
```



- Q should print the L from the topmost P on the stack.

Accessing Non-Local Variables...

```
PROCEDURE P (a:INTEGER);
  VAR L : INTEGER;
  PROCEDURE Q (x:INTEGER);
    BEGIN R(16) END Q;

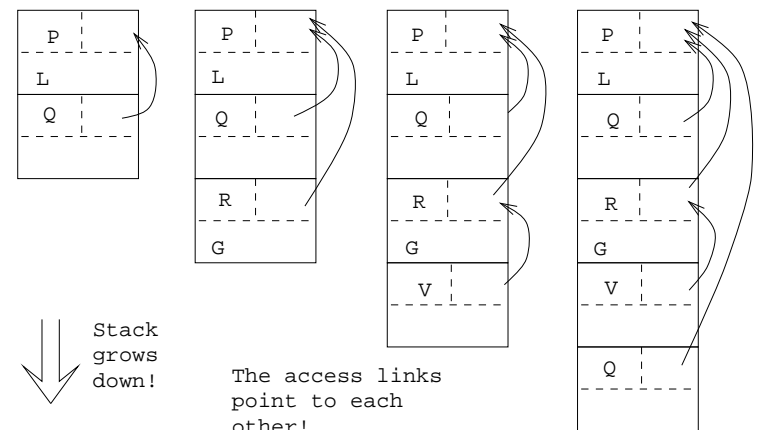
  PROCEDURE R (y:INTEGER);
    VAR G : INTEGER;
    PROCEDURE V (z:INTEGER);
      BEGIN Q(10) END V;
    BEGIN V(12) END R;

  BEGIN Q (5); END P;
```

Accessing Non-Local Variables...

```
PROCEDURE P (a:INTEGER);
  PROCEDURE Q (x:INTEGER);
  PROCEDURE R (y:INTEGER);
    PROCEDURE V (z:INTEGER);
```

- We give each activation record an **Access Link** (aka **Static Link**).
- Assume that Q is nested within P (as above). Then Q's static link points to the activation record for the most recent activation of P.



Accessing Non-Local Variables...

```

PROC P ();
  VAR L:INTEGER;           ⇐ nL = 1
  PROC R ();
    PROC V ();             nR - nL = 2
      BEGIN L:=...END V;   ⇐ nR = 3

```

Access to non-local variable L:

- Assume that L is declared at nesting level n_L , and that the reference to L is at nesting level n_R (as above).
- Follow $n_R - n_L$ access links. We now point to the activation record for the most recent activation of P.

Accessing Non-Local Variables...

```

PROC P ();
  VAR L:INTEGER;           ⇐ nL = 1
  PROC R ();
    PROC V ();             nR - nL = 2
      BEGIN L:=...END V;   ⇐ nR = 3

```

MIPS Example:

```

lw $2, AL($fp) # AL is offset of access link.
lw $2, ($2)    # An access link points to
               # the previous access link.
lw $3, 12($2)  # Get the data in the AR.

```

Setting up Access Links

- Every time we make a procedure call we have to set up the access link for the new procedure activation.
- There are two cases to consider:
 - when the callee is nested within the caller, and
 - when the caller is nested within the callee.

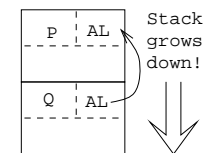
Setting up Access Links...

Case (1): Callee Within Caller:

```

PROC P ();           ⇐ NP = 1
  PROC Q ();         ⇐ NQ = 2
    PROC V ();
      ...
  BEGIN Q (); END P;

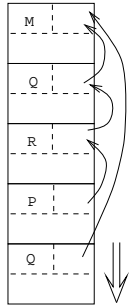
```



- P calls Q. P is at level N_P , Q is at level N_Q . $N_P = N_Q - 1$, since Q must be nested immediately within P.
- Make Q's access link point to the access link in P's activation record.

Case (2): Caller Within Callee:

```
PROG M;  
PROC Q();       $\Leftarrow N_Q = 1$   
  PROC R();  
    PROC P();   $\Leftarrow N_P = 3$   
  BEGIN  
    Q();  
  END P;  
END Q;
```



$$N_P - N_Q + 1 = 3$$

- P calls Q. P is at level N_P , Q is at level N_Q . $N_P \geq N_Q$.
- Traverse the access links to find the most recent activation of the first procedure which statically encloses both P and Q. We need to follow $N_P - N_Q + 1$ links.

- Read Scott, pp. 117–120, 408–412