Apendix I - Hierarchy.query
Query to Generate the Analyzer

```
ROOTPROC hierarchy
FILE dot “hier.dot”
PROC hierarchy
ROOT CFile;
{
LOCAL GNODE BASE;
LOCAL GNODE DERIVED;
<globals
  {Declaration
        [ (?ClassDef
            <defname (ASSIGN DERIVED $token)>
            <bases
               {BaseSpec
                  <refname (ASSIGN BASE $token)>
                  (PRINT dot “%s@%s” DERIVED BASE)
                  (PRINT dot “\n”)
               }
            >
        ) ]
  }
>
}
```

4. The Visualizer

The visualizer employed, DirVis, was developed for directory visualization by Mark Gray at the GVU Lab. For more information, he can be reached by email at `vatavian@cc.gatech.edu`.

The visualizer is called with the interface file using the command:

```
dirvis -h <interface filename>
```

The fullpath of DirVis is:

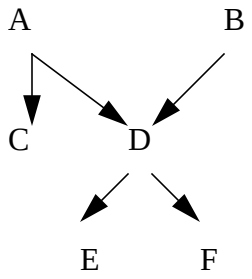
```
/net/gvu.1/public/bin/dirvis
```

5. Conclusions

The process involved in the development of this class visualizer was rather simple, what says a lot about the power of the tools involved. The analyzer generated sees robust and worked well not only with small programs but also with complex systems involving the inclusion of many standard and customized files. The key in those cases was to use the same makefiles employed for compilation but replacing the compiler name by the name of the script that invokes the analyzer. The documentation provided was clear and concise, but some features (like UNPARSE) were mentioned but not documented. Getting information about class containment, member functions, and other class attributes that should be available to a full-fleshed class hierarchy browser seems to be straightforward. Clearly the most challenging part would be designing the user-machine interaction and programming the visualization tool.

3. The Interface

For a class hierarchy like:



The output of the analyzer would be of the form:

C@A

D@A

D@B

E@D

F@D

To adapt that output to the input structure required by the visualizer two small C programs were written to generate the inheritance paths. The visualizer, designed for a tree-like directory structure expects linear sequences of paths and generates automatically a tree rooted at the interface file. Two approaches were implemented:

In the first approach each root's child is a base class without parents in the C++ class hierarchy. This allows immediate access to each subclass of a given class. The output for the example above would be:

xxxx E.D.A

xxxx F.D.A

xxxx C.A

In the alternative approach each root's child is a class without subclasses. Thus we can have immediate access to each base class of a given class. For this case each path in the output would be the reverse of the example above.

Building a Simple Class Hierarchy Browser with GEN++

D. Alberto Rama

1. Objective

This report summarizes how the prototype of a class hierarchy browser was built using GEN++ to generate an analyzer of C++ programs and adapting its output to interface with DirVis, a directory visualizer developed at the GVU Lab.

2. The Analyzer

GEN++ is a generator of analyzers for C++ programs based on GENOA, a language independent specification language and generation system developed by AT&T Labs. GEN++ essentially implements a query language for a semantic tree generated after parsing the program or set of programs to be analyzed. For each class declaration the base classes were determined in order to get a list of pairs (base class, derived class). App. I reproduces the query used in the generation of the analyzer, that was based on one example provided with the beta version of the package. The file with the query can be found at `jazz:/hs1/alber/gen++/pkg/hierarchy.query`, the example used as a base is on `jazz:/hs1/alber/gen++/pkg/examples/hier.query`.

The analyzer can be regenerated by typing:

```
mk hierarchy
```

(have to be logged onto jazz and the command should be run on directory `/hs1/alber/gen++/pkg`

The analyzer was renamed as `/hs1/alber/gen++/analyzers/hierarchy.`

More information on GEN++, the location of the documentation and how to generate and run analyzers can be found on the README files at directories `gen++`, `pkg` and `analyzers`.