Examples [A]

This appendix presents examples of some typical loads and segment tree structures with their corresponding sets of directives.

Basic case

A.1

The Fortran program in this example is compiled, loaded, and executed beginning at entry point START. The loader produces a full load map. Its source is in file `source.f`. The loaded program is nonsegmented.

```
cft77 source.f
segldr -o ftest -M,f -e START source.o > mapfile
ftest
```

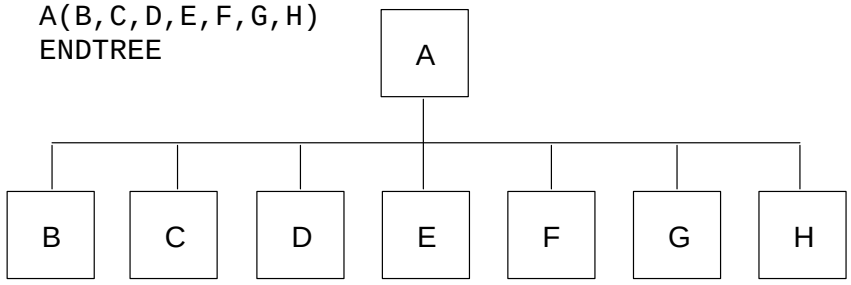
Tree structure examples

A.2

The following two examples show two legal tree structures generated by the loader.

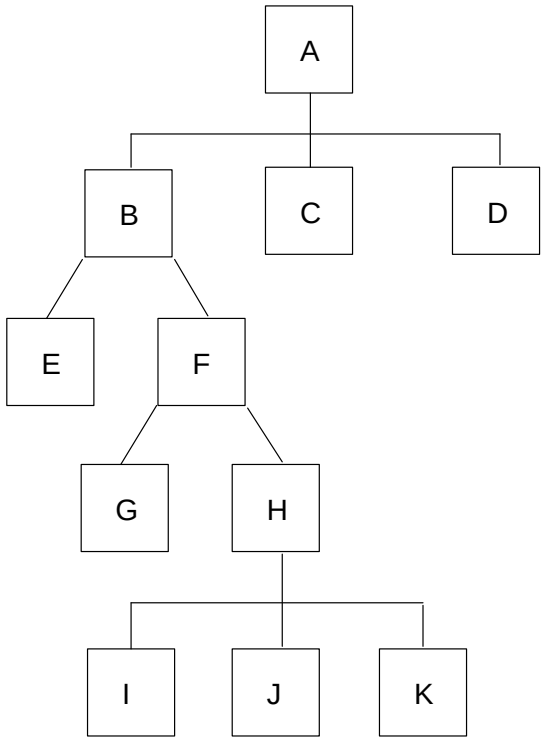
Example 1:

```
TREE
A(B,C,D,E,F,G,H)
ENDTREE
```



Example 2:

```
TREE
B(E,F)
H(I,J,K)
A(B,C,D)
F(G,H)
ENDTREE
```



Tree structure with expandable common block

A.3

Given the tree structure shown in Figure 12, assume that dynamic common block `/DYN/` is used and expanded at execution time. All modules are obtained from `mybin.o`, `blib.a`, and `baselib.a`. Common block `/AA/` is to be assigned to segment J. A full load map on file map is desired.

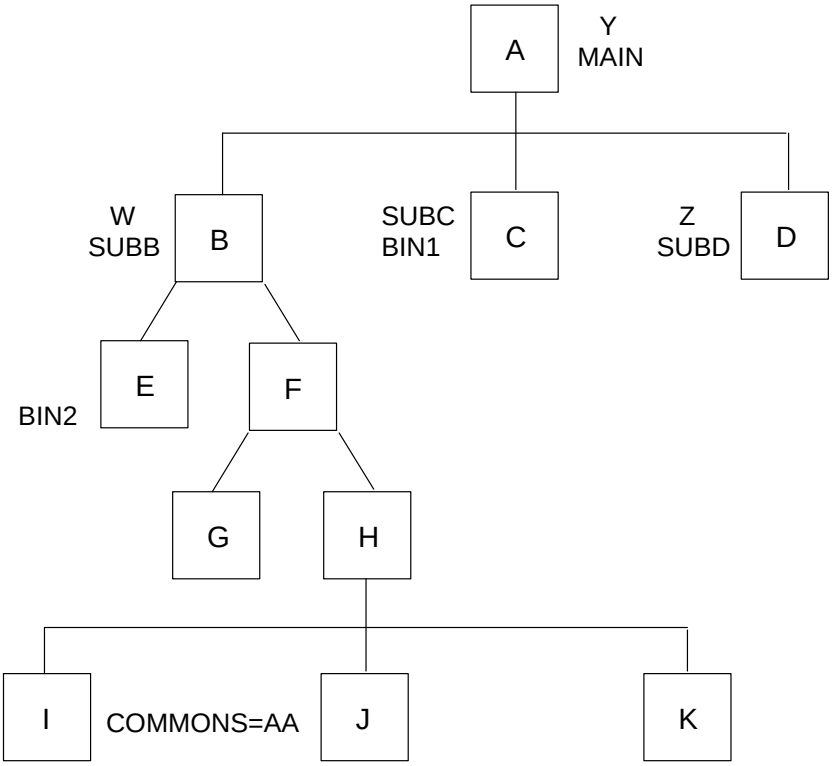


Figure 12. Example tree structure

The control statement and directives required are as follows:

```
segldr -i ins -M map,full -l./blib.a -l./baselib.a mybin.o
```

The following directives are used:

```
DYNAMIC=DYN
TREE
A(B,C,D)
B(E,F)
F(G,H)
H(I,J,K)
ENDTREE
SEGMENT=A
MODULES=MAIN
ENDSEG
SEGMENT=B
MODULES=SUBB
ENDSEG SEGMENT=C
MODULES=SUBC ENDSEG
SEGMENT=D
MODULES=SUBD
ENDSEG
SEGMENT=E
MODULES=SUBE
ENDSEG
SEGMENT=F
MODULES=SUBF
ENDSEG
SEGMENT=G
MODULES=SUBG
ENDSEG
SEGMENT=H
MODULES=SUBH
ENDSEG
SEGMENT=I
MODULES=SUBI
ENDSEG
SEGMENT=J
COMMONS=AA;MODULES=SUBJ
ENDSEG
SEGMENT=K
MODULES=SUBK
ENDSEG
```

Segmented load with duplicated modules

A.4

This example is based on the tree structure in Figure 13. Given this tree structure, assume that all modules in object file `bin1.o` are to be loaded in segment C and all modules in `bin2.o` in segment E. All other modules are to be obtained from global bin files `bin3.o` and `bin4.o`, and the default libraries. Modules Y, W, and Z are in segments A, B, and D, respectively. Also assume that segments B and C contain large data arrays whose updated values are needed each time they are executed. Assume that version 1 of module X (in `bin3.o`) is needed in segment D, and version 2 (in `bin4.o`) is needed in segment F. All calls to entry points Y1, Y2, and Y3 are to be linked to entry point Y. Also assume that the module name and the entry name in a subroutine are the same.

The control statements and directives included are as follows:

```
segldr -i inpts
```

`INPTS` contains the following directives:

```
BIN=bin3.o, bin4.o; EQUIV=Y (Y1,Y2,Y3)
TREE
A(B,C)
C(D,E,F)
ENDTREE
DUP=X(D,F)
SEGMENT=A
MODULES=Y
ENDSEG
SEGMENT=B;SAVE=ON
MODULES=W
ENDSEG
SEGMENT=C;SAVE=ON
BIN=bin1.o
ENDSEG
SEGMENT=D
MODULES=Z,X: bin3.o
ENDSEG
SEGMENT=E
BIN=bin2.o
ENDSEG
SEGMENT=F
MODULES=X:bin4.o
ENDSEG
```

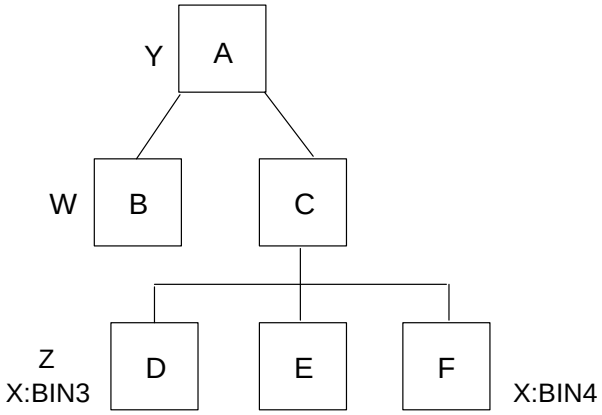


Figure 13. Tree structure

Comprehensive Fortran program example
A.5

This example provides a set of loader directives, block maps and associated output, and related entry point and common block reference maps for the sample Fortran program that follows.

Fortran source code
A.5.1

The following Fortran program consists of 10 subroutines. The loader directives described in the following subsection load the 10 separate modules of this program into separate segments.

```
PROGRAM EXAMPLE
DATA I /0/
CALL SUBR1(I)
CALL SUBR2(I)
PRINT *,' VALUE OF I IS ',I
END

SUBROUTINE SUBR1(I)
COMMON /SPACE/ SPACE(100)
COMMON COMMON
I=I+1
CALL SUBR1A(I)
CALL SUBR1B(I)
CALL SUBR1C(I)
RETURN
END
```

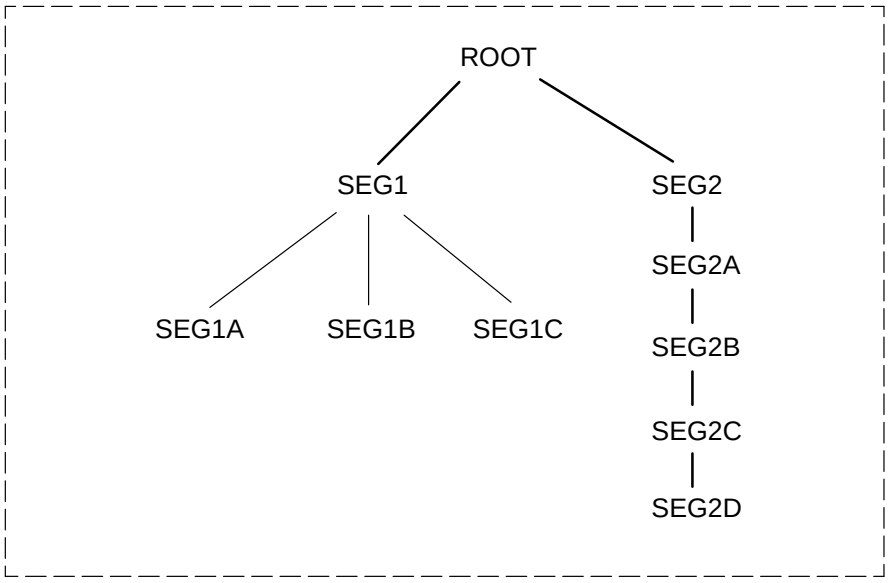
```
SUBROUTINE SUBR1A(I)
COMMON COMMON
PRINT *,' EXECUTION OF SUBR1A' I=I+1
RETURN
END

SUBROUTINE SUBR1B(I)
COMMON /STATUS/ STATUS
COMMON COMMON
PRINT *,' EXECUTION OF SUBR1B'
I=I+1
RETURN
END

SUBROUTINE SUBR1C(I)
COMMON /STATUS/ STATUS
PRINT *,' EXECUTION OF SUBR1C'
I=I+1
RETURN
END

SUBROUTINE SUBR2(I)
COMMON /SPACE/ SPACE(100)
I=I+1
CALL SUBR2A(I)
RETURN
END

SUBROUTINE SUBR2A(I)
PRINT *,' EXECUTION OF SUBR2A'
I=I+1
CALL SUBR2B(I)
RETURN
END

SUBROUTINE SUBR2B(I)
PRINT *,' EXECUTION OF SUBR2B'
I=I+1
CALL SUBR2C(I)
RETURN
END

SUBROUTINE SUBR2C(I)
PRINT *,' EXECUTION OF SUBR2C'
I=I+1
CALL SUBR2D(I)
RETURN
END
```

```
      SUBROUTINE SUBR2D(I)
      PRINT *,' EXECUTION OF SUBR2D'
      I=I+1
      RETURN
      END
```

Loader directives A.5.2

The following loader directive input sample specifies and diagrams the construction of the segmented object module.

```
ECHO=ON
MAP=FULL
HEAP=5000+0
```




```
TREE
      ROOT(SEG1,SEG2)
      SEG1(SEG1A,SEG1B,SEG1C)
      SEG2(SEG2A)
      SEG2A(SEG2B)
      SEG2B(SEG2C)
      SEG2C(SEG2D)
ENDTREE
SEGMENT=ROOT
      MODULES=EXAMPLE
ENDSEG
*
```

* *Left-hand segment tree branch*

```
*
SEGMENT=SEG1
      MODULES=SUBR1
ENDSEG
SEGMENT=SEG1A
      MODULES=SUBR1A
ENDSEG
SEGMENT=SEG1B
      MODULES=SUBR1B
ENDSEG
SEGMENT=SEG1C
      MODULES=SUBR1C
ENDSEG
*
* Right-hand segment tree branch
*
SEGMENT=SEG2
      MODULES=SUBR2
ENDSEG
SEGMENT=SEG2A
      MODULES=SUBR2A
ENDSEG SEGMENT=SEG2B
      MODULES=SUBR2B
ENDSEG
SEGMENT=SEG2C
      MODULES=SUBR2C
ENDSEG
SEGMENT=SEG2D
      MODULES=SUBR2D
ENDSEG
```

SEGLDR map output A.5.3

The following SEGLDR output sample is an example of the general information preceding the block maps. Word addresses and block lengths are in octal.

```
Program statistics
Segmented object module written to- a.out
Allocation order- XMP.EMA
Movable block positioning- ANY
Actual SLT requirement- 16
Program origin-           0 octal       0 decimal
Program length-      110403 octal   37123 decimal
Dynamic common block- //
        Origin-      110402 octal   37122 decimal
        Length-           1 octal       1 decimal
Maximum segment chain address-       110402 octal   37122 decimal
ending with segment- SEG2D
Transfer is to entry point- EXAMPLE at address- 340a

Managed Memory Statistics
   Initial stack size-                 4000 octal    2048 decimal
   Stack increment size-                400 octal     256 decimal
   Initial managed memory size-       11610 octal    5000 decimal
   Managed memory increment size-         0 octal       0 decimal
   Managed memory epsilon-                0 octal       0 decimal
   Base address of managed memory/stack-    76572
   Base address of pad area-                76367
Segment numbers
        0- ROOT     1- SEG1     2- SEG1A     3- SEG1B
        4- SEG1C    5- SEG2     6- SEG2A     7- SEG2B
        8- SEG2C    9- SEG2D
```

Program block maps A.5.4

The segment summary is followed by two block maps for each segment in this example; one sorted by address, and another sorted by block name. This is an abbreviated sample. Library routines have been omitted, and block maps for only the first three segments are present.

```
Segment Summary
Segment  Address   Length    Save Histogram (bar =– 884 words decimal)

ROOT           0    75763      ------------------------------------
SEG1        7576        46                                         –
SEG1A      76031        67                                         –
SEG1B      76031        67                                         –
SEG1C      76031         6                                         –
SEG2        7576        34                                         –
SEG2A      76017        73                                         –
SEG2B      76112        73                                         --
SEG2C      76205        73                                          –
SEG2D      76300        67                                          –

Segment 'ROOT' Block Map – sorted by address
Module   Block     Address   Length   Source       Date
$START                 0       22   /lib/libc.o  02/16/88 07:43
         TRBK         22        7
$SEGRES               31       73   /lib/libu.o  02/16/88 07:46
                     124       57
         CALLIST     203      115
         CALLIST     320        1
         TRBK        321       16
$EXAMPLE             337       27   t/example.o  01/22/87 16:16
         #TB         366        7
         #CL         375       22
         $TRBK       417        7
         /SPACE/   74365      144
         /WAVARS/  74531     1232                          (continued)
```

```
Segment 'ROOT' Block Map – sorted by block name
Module   Block     Address   Length   Source        Date

$SEGRES  CALLIST      320        1   /lib/libu.o   02/16/88 07:46
                      124       57
         /$SEGRES/  20633      462
                    21357      363
         CALLIST      203      115
         TRBK         321       16
                       31       73
$START               21317       40   /lib/libc.o   02/16/88 07:43
                         0       22
         TRBK           22        7
$EXAMPLE              337       27   t/example.o   01/22/87 16:16
         $TRBK        417        7
         #CL          375       22
         #TB          366        7
         #DA        61303        7

Segment 'SEG1' Block Map – sorted by address
Module   Block     Address   Length   Source        Date

SUBR1                75763       17   t/example.o   01/22/87 16:16
         #TB         76002        6
         #CL         76010        6
         $TRBK       76016        7
         #DA         76025        3
         /STATUS/    76030        1

 Segment 'SEG1' Block Map – sorted by block name
 Module  Block     Address   Length   Source        Date

         /STATUS/    76030        1
SUBR1    #CL         76010        6   t/example.o   01/22/87 16:16
         $TRBK       76016        7
         #DA         76025        3
         #TB         76002        6
                     75763       17                          (continued)
```

```
 Segment 'SEG1A' Block Map – sorted by address
 Module Block    Address   Length   Source       Date
 SUBR1A            76031       25   t/example.o  01/22/87 16:16
        #TB        76056       10
        #CL        76066       14
        $TRBK      76102        7
        #DA        76111        7

 Segment 'SEG1A' Block Map – sorted by block name
Module  Block    Address   Length   Source       Date

 SUBR1A #CL        76066       14   t/example.o  01/22/87 16:16
        #TB        76056       10
        $TRBK      76102        7
        #DA        76111        7
                   76031       25
```

Program entry-point cross-reference map
A.5.5

This sample entry-point cross-reference map shows entry-point values, segments to which modules are assigned, and the segment tree in caller/callee form.

When you specify `MAP=FULL` or `MAP=EPXRF`, this is the resulting output. (This sample is abbreviated for readability.)

```
Entry point references
EXAMPLE from t/example.o in ROOT     calls...  SUBR1 SUBR2 $WLI $WLA
                                               $WLV% $WLF $END
                                               $SEGCALL
    EXAMPLE          340a

 SUBR1 from t/example.o in SEG1      calls...  SUBR1A SUBR1B SUBR1C
    SUBR1          75764a   called by...EXAMPLE

SUBR1A from t/example.o in SEG1A     calls...  $WLI $WLA $WLF
    SUBR1A         76032a   called by...      SUBR1

SUBR1B from t/example.o in SEG1B     calls...  $WLI $WLA $WLF
    SUBR1B         76032a   called by...SUBR1

SUBR1C from t/example.o in SEG1C     calls...  $WLI $WLA $WLF
    SUBR1C         76032a   called by...SUBR1

SUBR2 from t/example.o in SEG2       calls...  SUBR2A
    SUBR2          75764a   called by...EXAMPLE

SUBR2A from t/example.o in SEG2A     calls...  $WLI $WLA $WLF SUBR2B
    SUBR2A         76020a   called by...SUBR2

SUBR2B from t/example.o in SEG2B     calls...  $WLI $WLA $WLF SUBR2C
    SUBR2B         76113a   called by...SUBR2A

SUBR2C from t/example.o in SEG2C     calls...  $WLI $WLA $WLF SUBR2C
    SUBR2C         76206a   called by...SUBR2B

SUBR2D from t/example.o in SEG2D     calls...  $WLI $WLA $WLF
    SUBR2D         76301a   called by...SUBR2C
```

Program common block reference map
A.5.6

When you specify `MAP=FULL` or `MAP=CBXRF`, the output contains the common block cross-reference.

```
Common Block References
Block    Segment  Address   Length   Module references
$SEGRES  ROOT       20633      462   $SEGRES

//                110402        1   SUBR1 SUBR1A SUBR1B

SPACE    ROOT       74365      144   SUBR1 SUBR2
STATUS   SEG1       76030        1   SUBR1B SUBR1C
```