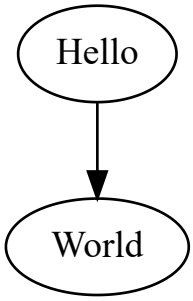
Examples

The following code examples are included in the `examples/` directory of the source repository/distribution. Most of them recreate examples from the graphviz.org gallery or the graphviz.org documentation.

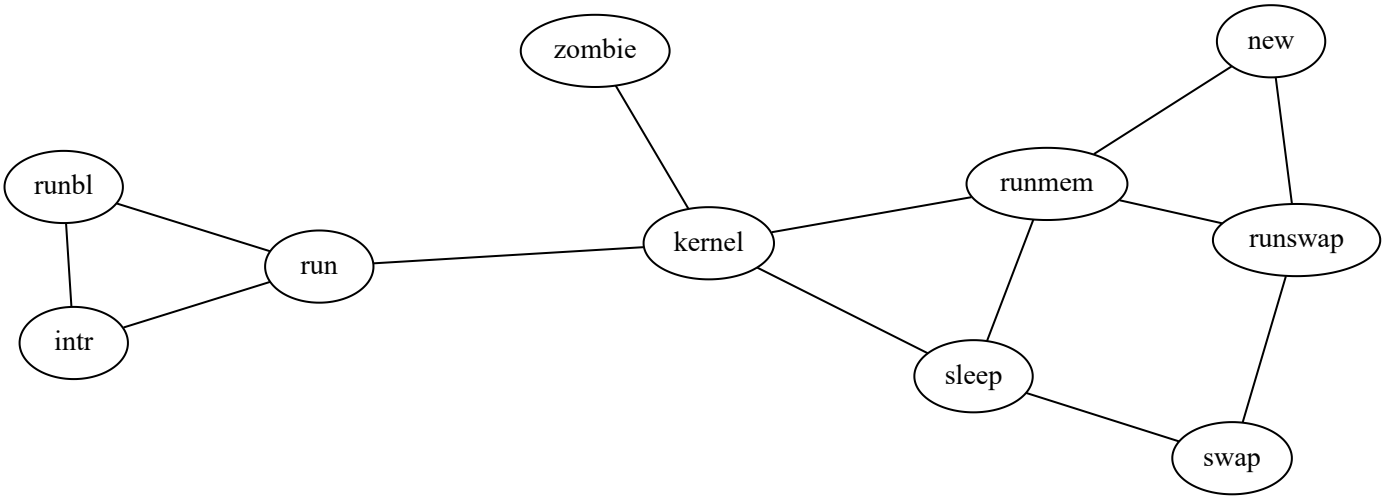
hello.py

```
# hello.py - http://www.graphviz.org/content/hello

from graphviz import Digraph

g = Digraph('G', filename='hello.gv')

g.edge('Hello', 'World')

g.view()
```



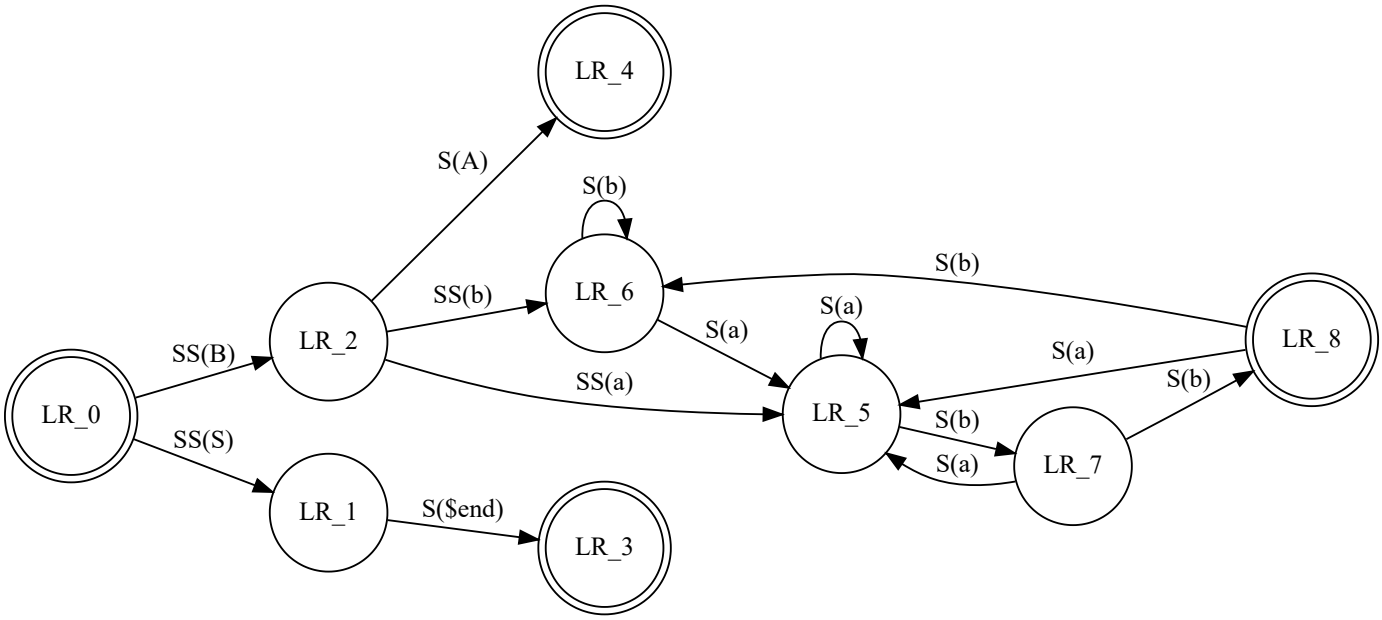
process.py

```
# process.py - http://www.graphviz.org/content/process

from graphviz import Graph

g = Graph('G', filename='process.gv', engine='sfdp')

g.edge('run', 'intr')
g.edge('intr', 'runbl')
g.edge('runbl', 'run')
g.edge('run', 'kernel')
g.edge('kernel', 'zombie')
g.edge('kernel', 'sleep')
g.edge('kernel', 'runmem')
g.edge('sleep', 'swap')
g.edge('swap', 'runswap')
g.edge('runswap', 'new')
g.edge('runswap', 'runmem')
g.edge('new', 'runmem')
g.edge('sleep', 'runmem')

g.view()
```



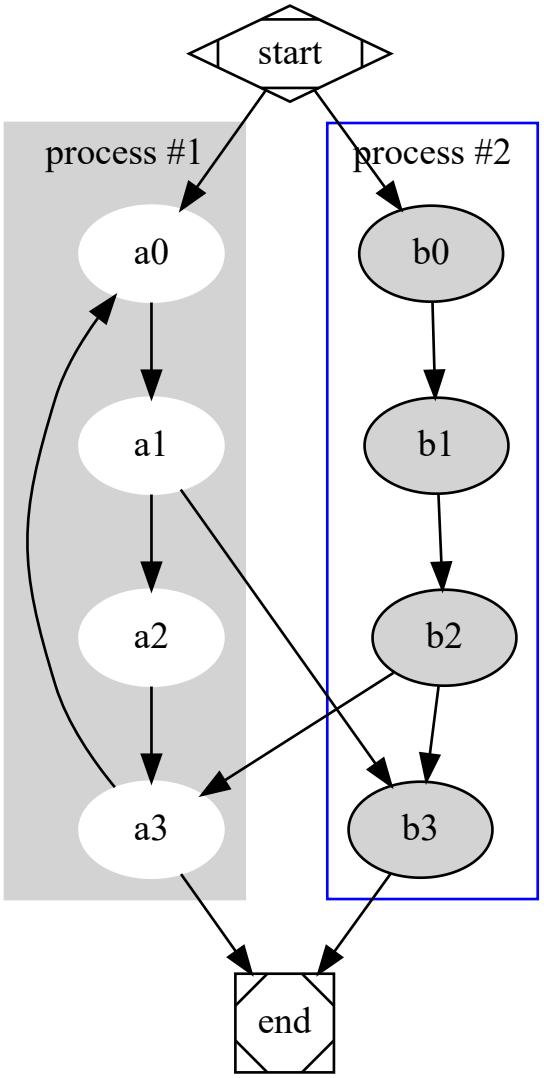
fsm.py

```
# fsm.py - http://www.graphviz.org/content/fsm

from graphviz import Digraph

f = Digraph('finite_state_machine', filename='fsm.gv')
f.attr(rankdir='LR', size='8,5')

f.attr('node', shape='doublecircle')
f.node('LR_0')
f.node('LR_3')
f.node('LR_4')
f.node('LR_8')

f.attr('node', shape='circle')
f.edge('LR_0', 'LR_2', label='SS(B)')
f.edge('LR_0', 'LR_1', label='SS(S)')
f.edge('LR_1', 'LR_3', label='S($end)')
f.edge('LR_2', 'LR_6', label='SS(b)')
f.edge('LR_2', 'LR_5', label='SS(a)')
f.edge('LR_2', 'LR_4', label='S(A)')
f.edge('LR_5', 'LR_7', label='S(b)')
f.edge('LR_5', 'LR_5', label='S(a)')
f.edge('LR_6', 'LR_6', label='S(b)')
f.edge('LR_6', 'LR_5', label='S(a)')
f.edge('LR_7', 'LR_8', label='S(b)')
f.edge('LR_7', 'LR_5', label='S(a)')
f.edge('LR_8', 'LR_6', label='S(b)')
f.edge('LR_8', 'LR_5', label='S(a)')

f.view()
```



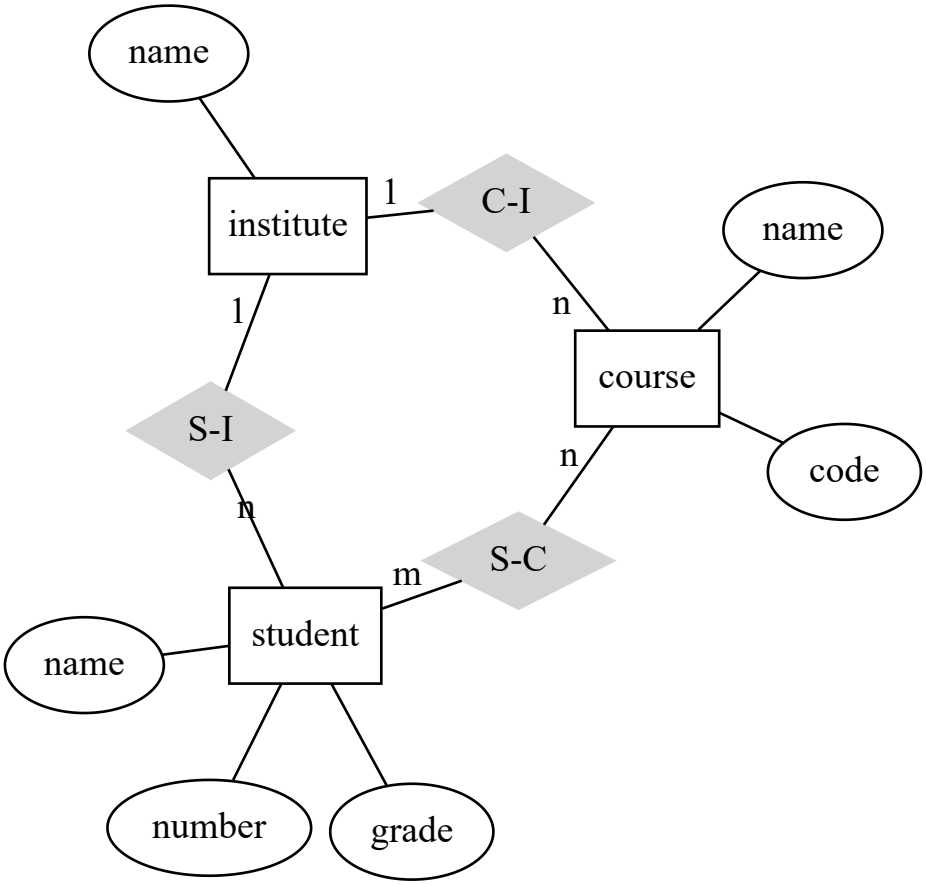
cluster.py

```
# cluster.py - http://www.graphviz.org/content/cluster

from graphviz import Digraph

g = Digraph('G', filename='cluster.gv')

# NOTE: the subgraph name needs to begin with 'cluster' (all lowercase)
#       so that Graphviz recognizes it as a special cluster subgraph

with g.subgraph(name='cluster_0') as c:
    c.attr(style='filled', color='lightgrey')
    c.node_attr.update(style='filled', color='white')
    c.edges([('a0', 'a1'), ('a1', 'a2'), ('a2', 'a3')])
    c.attr(label='process #1')

with g.subgraph(name='cluster_1') as c:
    c.attr(color='blue')
    c.node_attr['style'] = 'filled'
    c.edges([('b0', 'b1'), ('b1', 'b2'), ('b2', 'b3')])
    c.attr(label='process #2')

g.edge('start', 'a0')
g.edge('start', 'b0')
g.edge('a1', 'b3')
g.edge('b2', 'a3')
g.edge('a3', 'a0')
g.edge('a3', 'end')
g.edge('b3', 'end')

g.node('start', shape='Mdiamond')
g.node('end', shape='Msquare')

g.view()
```



er.py

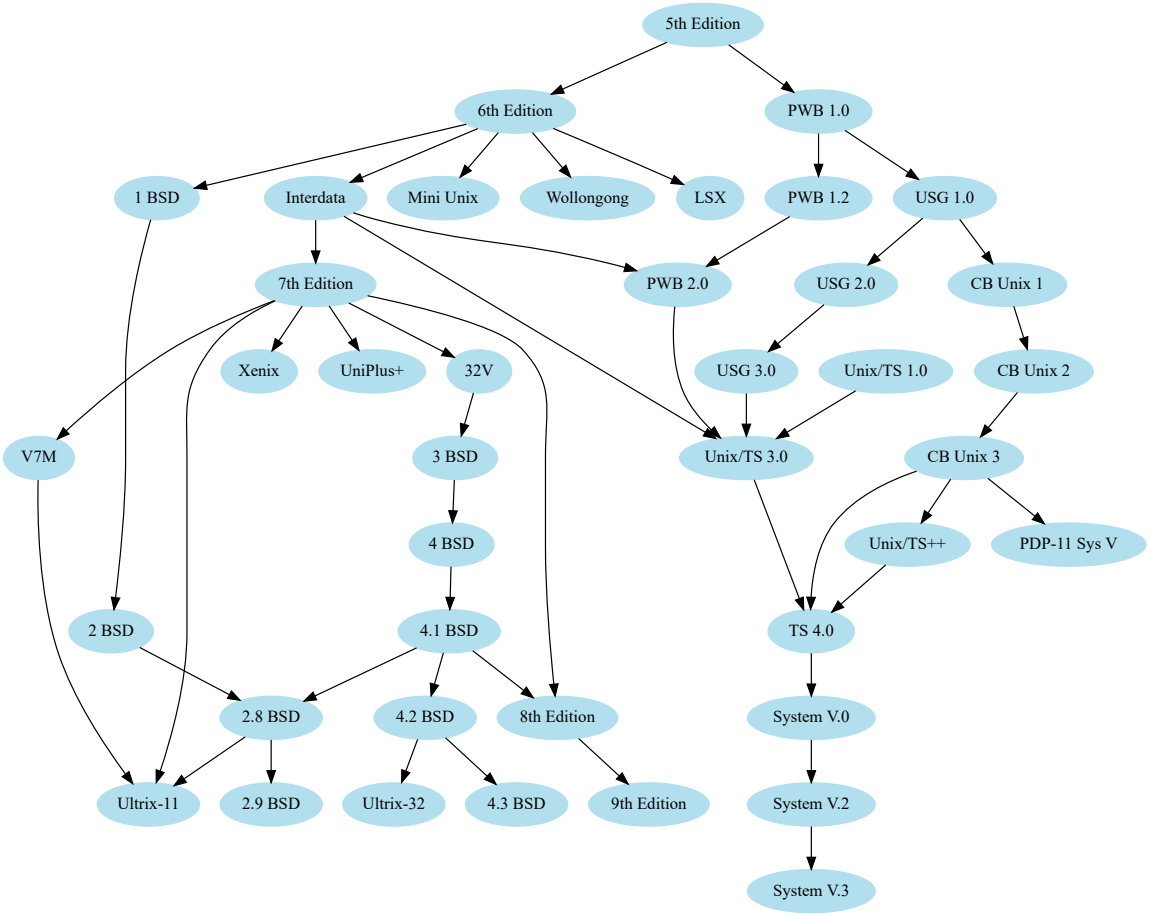
```
# er.py - http://www.graphviz.org/content/ER

from graphviz import Graph

e = Graph('ER', filename='er.gv', engine='neato')

e.attr('node', shape='box')
e.node('course')
e.node('institute')
e.node('student')

e.attr('node', shape='ellipse')
e.node('name0', label='name')
e.node('name1', label='name')
e.node('name2', label='name')
e.node('code')
e.node('grade')
e.node('number')

e.attr('node', shape='diamond', style='filled', color='lightgrey')
e.node('C-I')
e.node('S-C')
e.node('S-I')

e.edge('name0', 'course')
e.edge('code', 'course')
e.edge('course', 'C-I', label='n', len='1.00')
e.edge('C-I', 'institute', label='1', len='1.00')
e.edge('institute', 'name1')
e.edge('institute', 'S-I', label='1', len='1.00')
e.edge('S-I', 'student', label='n', len='1.00')
e.edge('student', 'grade')
e.edge('student', 'name2')
e.edge('student', 'number')
e.edge('student', 'S-C', label='m', len='1.00')
e.edge('S-C', 'course', label='n', len='1.00')

e.attr(label=r'\n\nEntity Relation Diagram\ndrawn by NEATO')
e.attr(fontsize='20')

e.view()
```



Entity Relation Diagram
drawn by NEATO

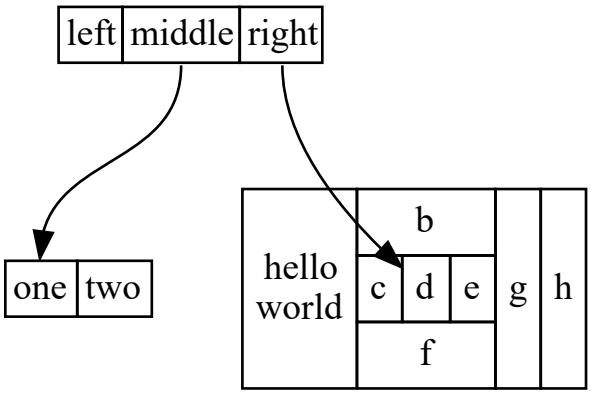
unix.py

```
# unix.py - http://www.graphviz.org/content/unix

from graphviz import Digraph

u = Digraph('unix', filename='unix.gv',
            node_attr={'color': 'lightblue2', 'style': 'filled'})
u.attr(size='6,6')

u.edge('5th Edition', '6th Edition')
u.edge('5th Edition', 'PWB 1.0')
u.edge('6th Edition', 'LSX')
u.edge('6th Edition', '1 BSD')
u.edge('6th Edition', 'Mini Unix')
u.edge('6th Edition', 'Wollongong')
u.edge('6th Edition', 'Interdata')
u.edge('Interdata', 'Unix/TS 3.0')
u.edge('Interdata', 'PWB 2.0')
u.edge('Interdata', '7th Edition')
u.edge('7th Edition', '8th Edition')
u.edge('7th Edition', '32V')
u.edge('7th Edition', 'V7M')
u.edge('7th Edition', 'Ultrix-11')
u.edge('7th Edition', 'Xenix')
u.edge('7th Edition', 'UniPlus+')
u.edge('V7M', 'Ultrix-11')
u.edge('8th Edition', '9th Edition')
u.edge('1 BSD', '2 BSD')
u.edge('2 BSD', '2.8 BSD')
u.edge('2.8 BSD', 'Ultrix-11')
u.edge('2.8 BSD', '2.9 BSD')
u.edge('32V', '3 BSD')
u.edge('3 BSD', '4 BSD')
u.edge('4 BSD', '4.1 BSD')
u.edge('4.1 BSD', '4.2 BSD')
u.edge('4.1 BSD', '2.8 BSD')
u.edge('4.1 BSD', '8th Edition')
u.edge('4.2 BSD', '4.3 BSD')
u.edge('4.2 BSD', 'Ultrix-32')
u.edge('PWB 1.0', 'PWB 1.2')
u.edge('PWB 1.0', 'USG 1.0')
u.edge('PWB 1.2', 'PWB 2.0')
u.edge('USG 1.0', 'CB Unix 1')
u.edge('USG 1.0', 'USG 2.0')
u.edge('CB Unix 1', 'CB Unix 2')
u.edge('CB Unix 2', 'CB Unix 3')
u.edge('CB Unix 3', 'Unix/TS++')
u.edge('CB Unix 3', 'PDP-11 Sys V')
u.edge('USG 2.0', 'USG 3.0')
u.edge('USG 3.0', 'Unix/TS 3.0')
u.edge('PWB 2.0', 'Unix/TS 3.0')
u.edge('Unix/TS 1.0', 'Unix/TS 3.0')
u.edge('Unix/TS 3.0', 'TS 4.0')
u.edge('Unix/TS++', 'TS 4.0')
u.edge('CB Unix 3', 'TS 4.0')
u.edge('TS 4.0', 'System V.0')
u.edge('System V.0', 'System V.2')
u.edge('System V.2', 'System V.3')

u.view()
```


5th Edition
6th Edition
PWB 1.0
1 BSD
Interdata
Mini Unix
Wollongong
LSX
PWB 1.2
USG 1.0
7th Edition
PWB 2.0
USG 2.0
CB Unix 1
Xenix
UniPlus+
32V
USG 3.0
Unix/TS 1.0
CB Unix 2
V7M
3 BSD
Unix/TS 3.0
CB Unix 3
4 BSD
Unix/TS++
PDP-11 Sys V
2 BSD
4.1 BSD
TS 4.0
2.8 BSD
4.2 BSD
8th Edition
System V.0
Ultrix-11
2.9 BSD
Ultrix-32
4.3 BSD
9th Edition
System V.2
System V.3

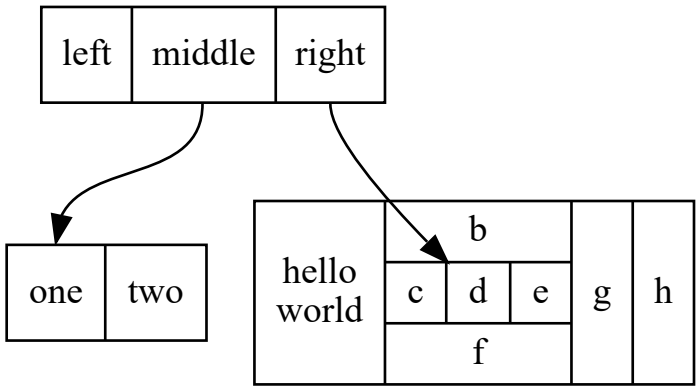
structs.py

```
# structs.py - http://www.graphviz.org/doc/info/shapes.html#html

from graphviz import Digraph

s = Digraph('structs', node_attr={'shape': 'plaintext'})

s.node('struct1', '''<
<TABLE BORDER="0" CELLBORDER="1" CELLSPACING="0">
  <TR>
    <TD>left</TD>
    <TD PORT="f1">middle</TD>
    <TD PORT="f2">right</TD>
  </TR>
</TABLE>>''')
s.node('struct2', '''<
<TABLE BORDER="0" CELLBORDER="1" CELLSPACING="0">
  <TR>
    <TD PORT="f0">one</TD>
    <TD>two</TD>
  </TR>
</TABLE>>''')
s.node('struct3', '''<
<TABLE BORDER="0" CELLBORDER="1" CELLSPACING="0" CELLPADDING="4">
  <TR>
    <TD ROWSPAN="3">hello<BR/>world</TD>
    <TD COLSPAN="3">b</TD>
    <TD ROWSPAN="3">g</TD>
    <TD ROWSPAN="3">h</TD>
  </TR>
  <TR>
    <TD>c</TD>
    <TD PORT="here">d</TD>
    <TD>e</TD>
  </TR>
  <TR>
    <TD COLSPAN="3">f</TD>
  </TR>
</TABLE>>''')

s.edges([('struct1:f1', 'struct2:f0'), ('struct1:f2', 'struct3:here')])

s.view()
```



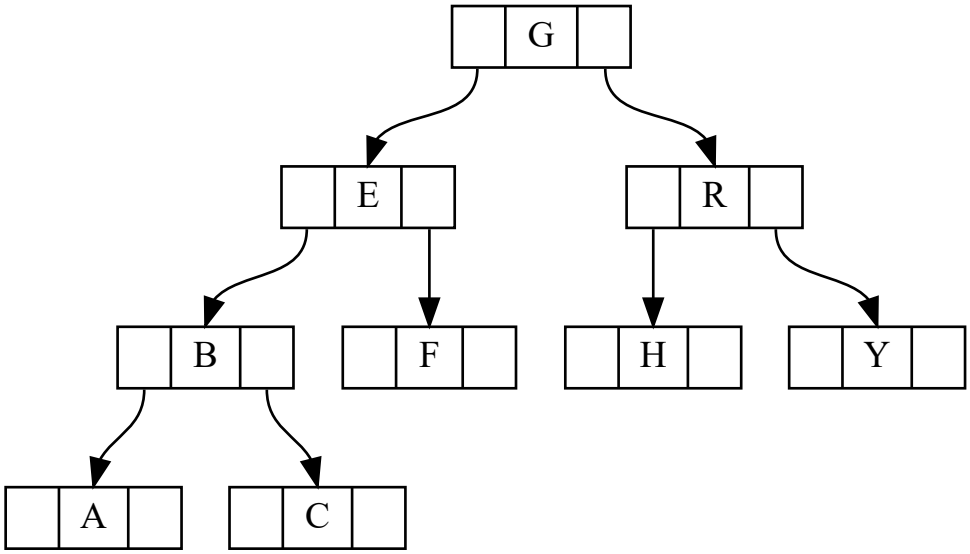
structs_revisited.py

```
# structs_revisited.py - http://www.graphviz.org/pdf/dotguide.pdf Figure 12

from graphviz import Digraph

s = Digraph('structs', filename='structs_revisited.gv',
            node_attr={'shape': 'record'})

s.node('struct1', '<f0> left|<f1> middle|<f2> right')
s.node('struct2', '<f0> one|<f1> two')
s.node('struct3', r'hello\nworld |{ b |{c|<here> d|e}| f}| g | h')

s.edges([('struct1:f1', 'struct2:f0'), ('struct1:f2', 'struct3:here')])

s.view()
```



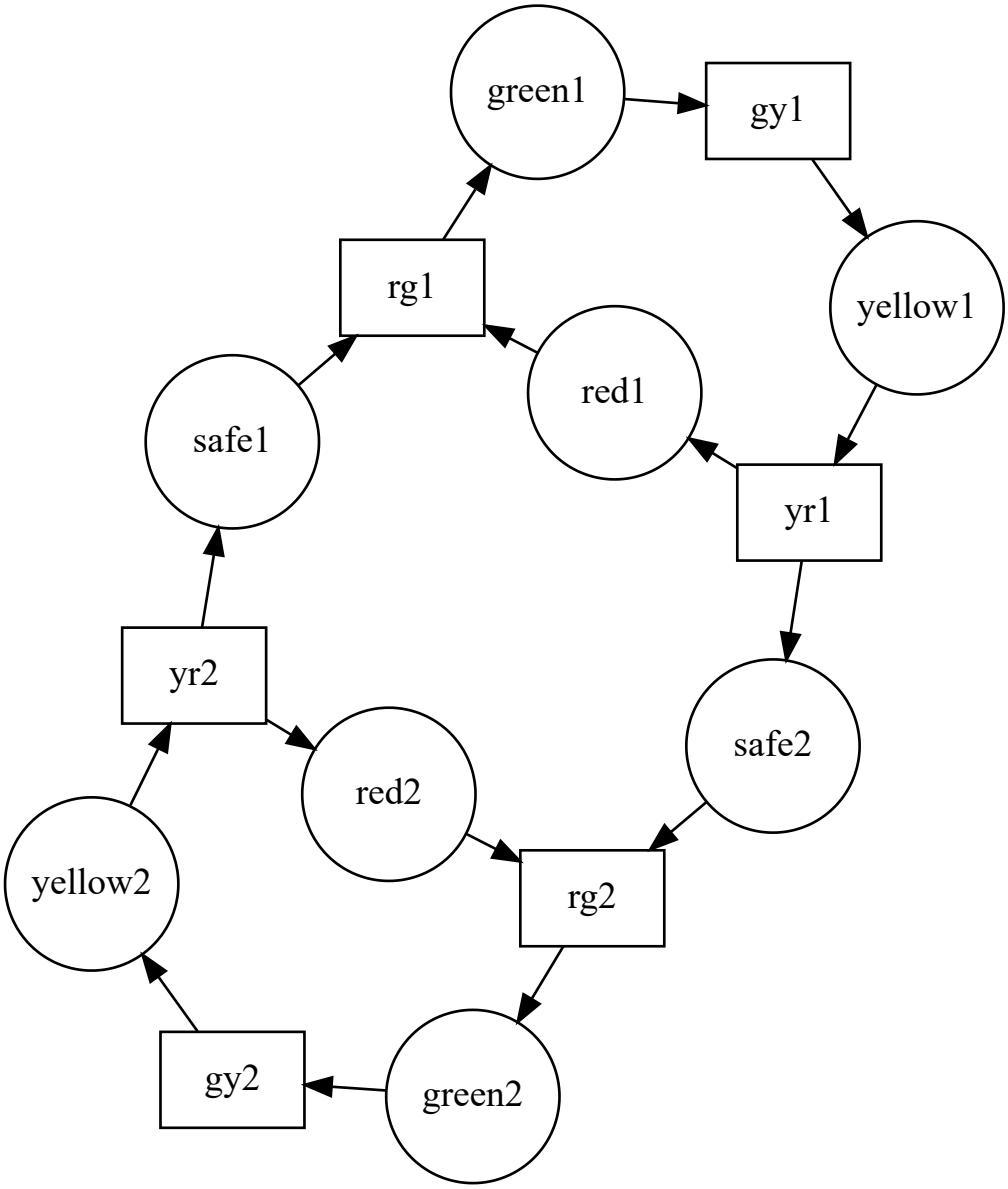
btree.py

```
# btree.py - http://www.graphviz.org/pdf/dotguide.pdf Figure 13

from graphviz import Digraph, nohtml

g = Digraph('g', filename='btree.gv',
            node_attr={'shape': 'record', 'height': '.1'})

g.node('node0', nohtml('<f0> |<f1> G|<f2>'))
g.node('node1', nohtml('<f0> |<f1> E|<f2>'))
g.node('node2', nohtml('<f0> |<f1> B|<f2>'))
g.node('node3', nohtml('<f0> |<f1> F|<f2>'))
g.node('node4', nohtml('<f0> |<f1> R|<f2>'))
g.node('node5', nohtml('<f0> |<f1> H|<f2>'))
g.node('node6', nohtml('<f0> |<f1> Y|<f2>'))
g.node('node7', nohtml('<f0> |<f1> A|<f2>'))
g.node('node8', nohtml('<f0> |<f1> C|<f2>'))

g.edge('node0:f2', 'node4:f1')
g.edge('node0:f0', 'node1:f1')
g.edge('node1:f0', 'node2:f1')
g.edge('node1:f2', 'node3:f1')
g.edge('node2:f2', 'node8:f1')
g.edge('node2:f0', 'node7:f1')
g.edge('node4:f2', 'node6:f1')
g.edge('node4:f0', 'node5:f1')

g.view()
```



traffic_lights.py

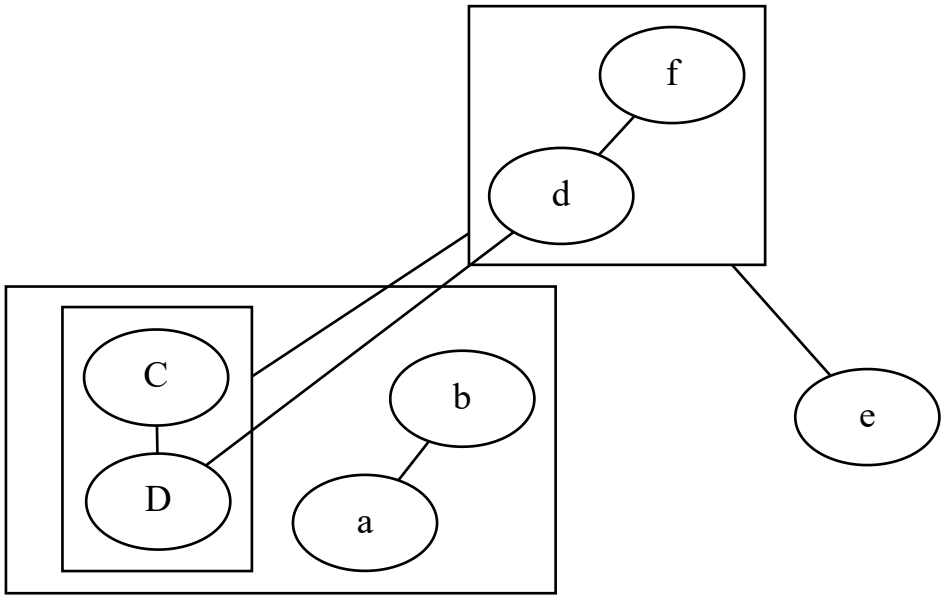
```
# traffic_lights.py - http://www.graphviz.org/content/traffic_lights

from graphviz import Digraph

t = Digraph('TrafficLights', filename='traffic_lights.gv', engine='neato')

t.attr('node', shape='box')
for i in (2, 1):
    t.node('gy%d' % i)
    t.node('yr%d' % i)
    t.node('rg%d' % i)

t.attr('node', shape='circle', fixedsize='true', width='0.9')
for i in (2, 1):
    t.node('green%d' % i)
    t.node('yellow%d' % i)
    t.node('red%d' % i)
    t.node('safe%d' % i)

for i, j in [(2, 1), (1, 2)]:
    t.edge('gy%d' % i, 'yellow%d' % i)
    t.edge('rg%d' % i, 'green%d' % i)
    t.edge('yr%d' % i, 'safe%d' % j)
    t.edge('yr%d' % i, 'red%d' % i)
    t.edge('safe%d' % i, 'rg%d' % i)
    t.edge('green%d' % i, 'gy%d' % i)
    t.edge('yellow%d' % i, 'yr%d' % i)
    t.edge('red%d' % i, 'rg%d' % i)

t.attr(overlap='false')
t.attr(label=r'PetriNet Model TrafficLights\n'
             r'Extracted from ConceptBase and layed out by Graphviz')
t.attr(fontsize='12')

t.view()
```



PetriNet Model TrafficLights
Extracted from ConceptBase and layed out by Graphviz

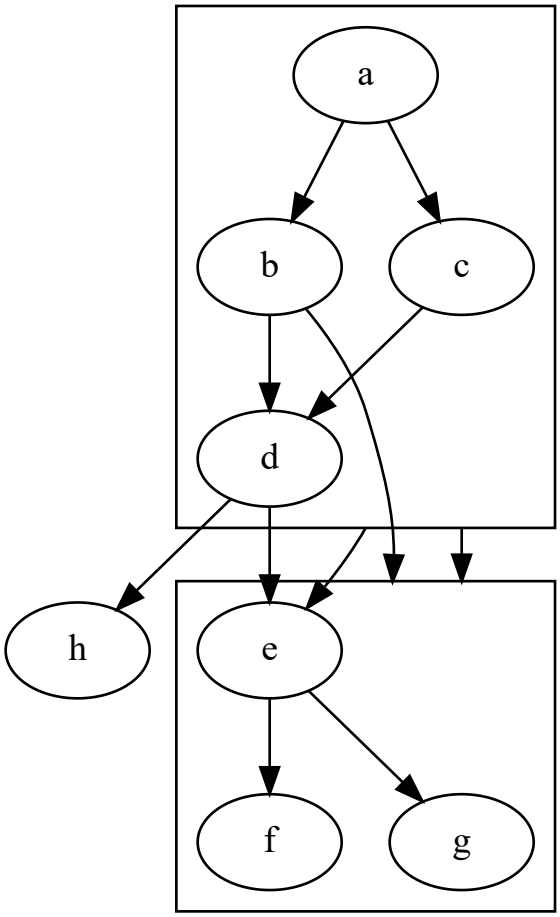
fdpclust.py

```
# fdpclust.py - http://www.graphviz.org/content/fdpclust

from graphviz import Graph

g = Graph('G', filename='fdpclust.gv', engine='fdp')

g.node('e')

with g.subgraph(name='clusterA') as a:
    a.edge('a', 'b')
    with a.subgraph(name='clusterC') as c:
        c.edge('C', 'D')

with g.subgraph(name='clusterB') as b:
    b.edge('d', 'f')

g.edge('d', 'D')
g.edge('e', 'clusterB')
g.edge('clusterC', 'clusterB')

g.view()
```



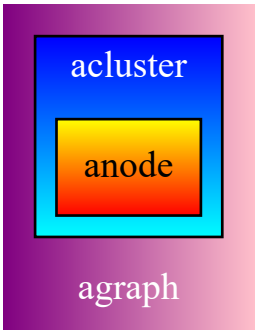
cluster_edge.py

```
# cluster_edge.py - http://www.graphviz.org/pdf/dotguide.pdf Figure 20

from graphviz import Digraph

g = Digraph('G', filename='cluster_edge.gv')
g.attr(compound='true')

with g.subgraph(name='cluster0') as c:
    c.edges(['ab', 'ac', 'bd', 'cd'])

with g.subgraph(name='cluster1') as c:
    c.edges(['eg', 'ef'])

g.edge('b', 'f', lhead='cluster1')
g.edge('d', 'e')
g.edge('c', 'g', ltail='cluster0', lhead='cluster1')
g.edge('c', 'e', ltail='cluster0')
g.edge('d', 'h')

g.view()
```



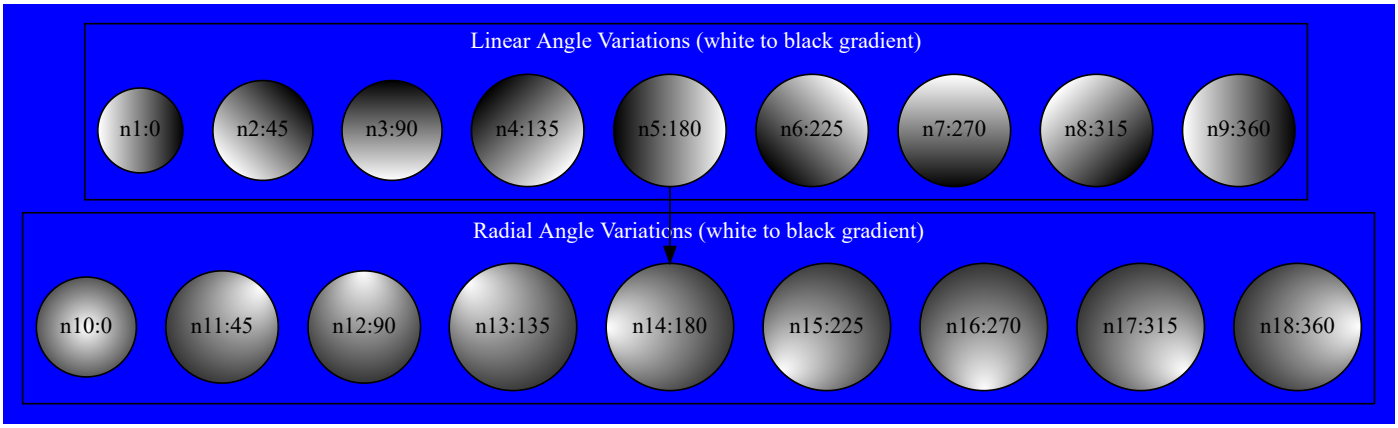
g_c_n.py

```
# http://www.graphviz.org/Gallery/gradient/g_c_n.html

from graphviz import Graph

g = Graph('G', filename='g_c_n.gv')
g.attr(bgcolor='purple:pink', label='agraph', fontcolor='white')

with g.subgraph(name='cluster1') as c:
    c.attr(fillcolor='blue:cyan', label='acluster', fontcolor='white',
           style='filled', gradientangle='270')
    c.attr('node', shape='box', fillcolor='red:yellow',
           style='filled', gradientangle='90')
    c.node('anode')

g.view()
```



angles.py

```
# angles.py - http://www.graphviz.org/Gallery/gradient/angles.html

from graphviz import Digraph

g = Digraph('G', filename='angles.gv')
g.attr(bgcolor='blue')

with g.subgraph(name='cluster_1') as c:
    c.attr(fontcolor='white')
    c.attr('node', shape='circle', style='filled', fillcolor='white:black',
           gradientangle='360', label='n9:360', fontcolor='black')
    c.node('n9')
    for i, a in zip(range(8, 0, -1), range(360 - 45, -1, -45)):
        c.attr('node', gradientangle='%d' % a, label='n%d:%d' % (i, a))
        c.node('n%d' % i)
    c.attr(label='Linear Angle Variations (white to black gradient)')

with g.subgraph(name='cluster_2') as c:
    c.attr(fontcolor='white')
    c.attr('node', shape='circle', style='radial', fillcolor='white:black',
           gradientangle='360', label='n18:360', fontcolor='black')
    c.node('n18')
    for i, a in zip(range(17, 9, -1), range(360 - 45, -1, -45)):
        c.attr('node', gradientangle='%d' % a, label='n%d:%d' % (i, a))
        c.node('n%d' % i)
    c.attr(label='Radial Angle Variations (white to black gradient)')

g.edge('n5', 'n14')

g.view()
```



rank_same.py

```
# https://stackoverflow.com/questions/25734244/how-do-i-place-nodes-on-the-same-level-in-dot

import graphviz

d = graphviz.Digraph(filename='rank_same.gv')

with d.subgraph() as s:
    s.attr(rank='same')
    s.node('A')
    s.node('X')

d.node('C')

with d.subgraph() as s:
    s.attr(rank='same')
    s.node('B')
    s.node('D')
    s.node('Y')

d.edges(['AB', 'AC', 'CD', 'XY'])

d.view()
```