NAME

prune – Prune directed graphs

SYNOPSIS

prune [**–n** *node*] [**–N** *attrspec*] [**–v**] [**–(h|?)**] [*files ...*]

DESCRIPTION

prune reads directed graphs in the same format used by **dot(1)** and removes subgraphs rooted at nodes specified on the command line via options. These nodes themselves will not be removed, but can be given attributes so that they can be easily located by a graph stream editor such as **gvpr(1).** **prune** correctly handles cycles, loops and multi-edges.

Both options can appear multiple times on the command line. All subgraphs rooted at the respective nodes given will then be processed. If a node does not exist, **prune** will skip it and print a warning message to stderr. If multiple attributes are given, they will be applied to all nodes that have been processed. **prune** writes the result to the stdout.

OPTIONS

–n *name*
: Specifies name of node to prune.

–N *attrspec*
: Specifies attribute that will be set (or changed if it exists) for any pruned node. *attrspec* is a string of the form *attr=value.*

–v
: Verbose output.

–h –?
: Prints the usage and exits.

EXAMPLES

An input graph *test.gv* of the form

```
digraph DG {
  A –> B;
  A –> C;

  B –> D;
  B –> E;
}
```

, processed by the command

```
prune –n B test.gv
```

would produce the following output (the actual code might be formatted in a slightly different way).

```
digraph DG {
  A –> B;
  A –> C;
}
```

Another input graph *test.gv* of the form

```
digraph DG {
  A –> B;
  A –> C;

  B –> D;
  B –> E;

  C –> E;
}
```

(note the additional edge from *C* to *E*), processed by the command

```
prune −n B −N color=red test.gv
```

results in

```
digraph DG {
  B [color=red];
  A −> B;
  A −> C;
  C −> E;
}
```

Node *E* has not been removed since its second parent *C* is not being pruned.

EXIT STATUS

prune returns 0 on successful completion. It returns 1 if an error occurs.

SEE ALSO

dot(1), **gvpr**(1)

AUTHOR

Marcus Harnisch <marcus.harnisch@gmx.net>