SPACE APPLICATION NOTE ABOUT UNINITIALIZED PREVIOUS POINTER

S. de Graaf

Circuits and Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
Delft University of Technology
The Netherlands

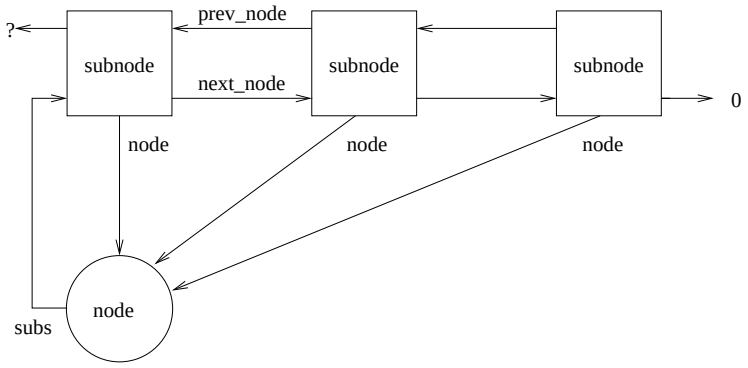
Report EWI-ENS 03-07
October 27, 2003

Last revision: October 27, 2003.

1. INTRODUCTION

The uninitialized previous pointer technique can be used for data structs in the space program. Because this technique is maybe unknown to space programmers, i shall explain this software technique in this application node.

The uninitialized previous pointer technique can be used by double linked lists. As an example is used the double linked list between subnodes and nodes, see the following figure.



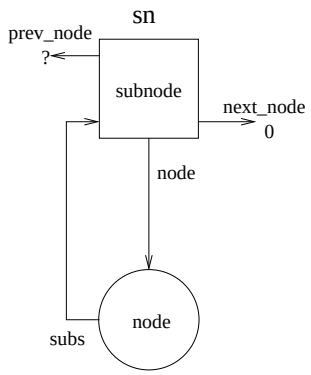
The first previous pointer "prev_node" needs not to be initialized.
I shall explain this software technique on the hand of the following four functions:

1. subnodeNew
2. subnodeCopy
3. subnodeJoin
4. subnodeDel

2. FUNCTION subnodeNew

The subnodeNew function initializes a new subnode "sn" and creates a node. A subnode must always point to a node. The subnode next pointer "next_node" is initialized to NULL. The previous pointer "prev_node" is not initialized. The node "subs" pointer points to the head of the subnode list.

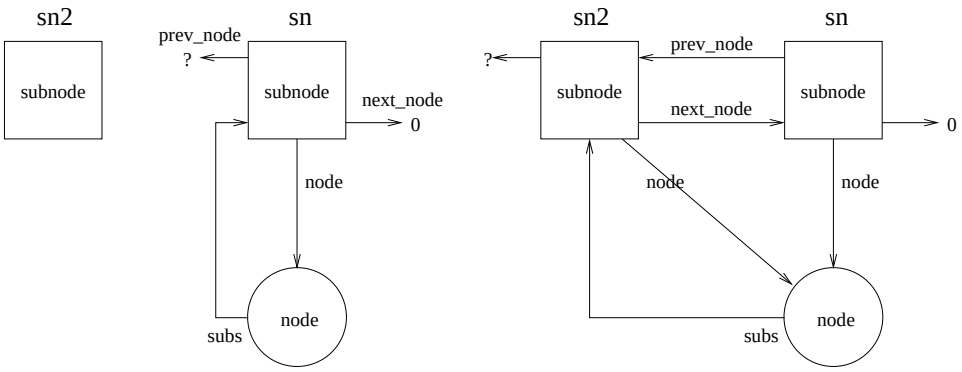
```
subnodeNew (sn)
{
    sn -> next_node = NULL;
    sn -> node = createNode();
    sn -> node -> subs = sn;
}
```



3. FUNCTION subnodeCopy

The subnodeCopy function initializes a second new subnode "sn2" and adds it to the subnode list of sn->node. Subnode "sn2" becomes the head of the subnode list. The "sn2" previous pointer "prev_node" is not initialized.

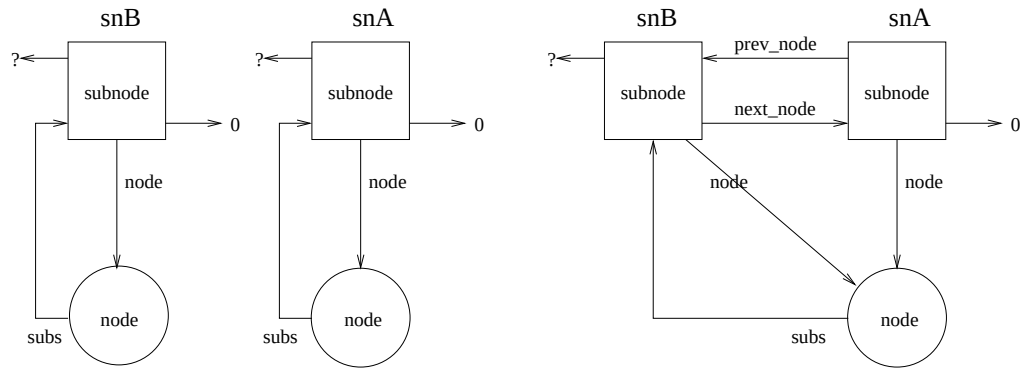
```
subnodeCopy (sn, sn2)
{
    sn2 -> next_node = sn -> node -> subs;
    sn2 -> next_node -> prev_node = sn2;
    sn2 -> node = sn -> node;
    sn2 -> node -> subs = sn2;
}
```



4. FUNCTION subnodeJoin

Function subnodeJoin joins the subnode lists of both nodes. The subnodes of the snB list must point to snA->node, because snB->node is deleted. This function does not use the "prev_node" pointers.

```
subnodeJoin (snA, snB)
{
    if (snA -> node != snB -> node) {
      for (sn = snB -> node -> subs; ; sn = sn -> next_node) {
          sn -> node = snA -> node;
          if (!sn -> next_node) break;
      }
      sn -> next_node = snA -> node -> subs;
      snA -> node -> subs -> prev_node = sn;
      snA -> node -> subs = snB -> node -> subs;
      nodeDel (snB -> node);
    }
}
```



5. FUNCTION subnodeDel

Function subnodeDel removes subnode "sn" from the subnode list of sn->node. When the subnode is the head of the list and there is not a next subnode, then the node is ready and can possible be deleted. This function uses only the "prev_node" pointer, when the subnode is not the head of the subnode list.

```
subnodeDel (sn)
{
    if (sn -> node -> subs == sn) {
      if (sn -> next_node)
          sn -> node -> subs = sn -> next_node;
      else
          readyNode (sn -> node);
    }
    else {
      sn -> prev_node -> next_node = sn -> next_node;
      if (sn -> next_node)
          sn -> next_node -> prev_node = sn -> prev_node;
    }
}
```