
Subject: linked list example

Posted by [Doug Larson](#) on Wed, 02 Sep 1998 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

Here is a set of linked list modules I made for my own use. I hesitate to post them since they are not "bullet-proof" but they do the job. Hopefully someone will be inspired to write a better open source linked list library.

My system administrator would not buy the "Fanning" book for me due to tight resources, so I had to invent my own (kluge) solutions.

Cheers,
Douglas J. Larson, PiledHigher&Deeper
Space Radiation Lab
California Institute of Technology
Mail Code: 220-47
Pasadena, CA 91125

```
. *****  
;  
;+  
; Setup_llist.pro  
;  
; PURPOSE:  
;   Setup a new node of a generic linked list.  
;  
; CATEGORY:  
;   Generic linked list generator. This function is NOT an insert operator.  
;  
; CALLING SEQUENCE:  
; somenewnode = ptr_new(Setup_llist(bugmsg))  
;  
; KEYWORD PARAMETERS:  
;  
; COMMON BLOCKS:  
; None  
;  
; OUTPUTS:  
; Data structure  
;  
; MODIFICATION HISTORY:  
; Created by: Douglas J. Larson, 18 August, 1998.  
;-  
. *****  
;
```

```

FUNCTION Setup_llist, bugmsg

if ( bugmsg eq 1 ) then message, ' Entered', /INFORMATIONAL

lls = { nodePtr:PTR_NEW(), $ ; Points to the linked list information
       next:PTR_NEW() } ; Pointer to the next data vector

if ( bugmsg eq 1 ) then message, ' Exited', /INFORMATIONAL

RETURN, lls

END
. *****
;+
; llist_insert.pro
;
; PURPOSE:
;   Setup a new node of a generic linked list.
;
; CATEGORY:
;   Generic linked list generator. This function is NOT an insert operator.
;
; CALLING SEQUENCE:
;
; KEYWORD PARAMETERS:
;
; COMMON BLOCKS:
; None
;
; OUTPUTS:
; Data structure
;
; MODIFICATION HISTORY:
; Created by: Douglas J. Larson, 18 August, 1998.
;-
. *****
FUNCTION llist_insert, bugmsg, startPtr, value

if ( bugmsg eq 1 ) then message, ' Entered', /INFORMATIONAL

if NOT(PTR_VALID(startPtr)) then begin
  if ( bugmsg eq 1 ) then message, 'startPtr is NULL', /INFORMATIONAL
  startPtr = PTR_NEW(Setup_llist(bugmsg))
  currentPtr = startPtr
endif else if (PTR_VALID(startPtr)) then begin
  if ( bugmsg eq 1 ) then message, 'startPtr is Valid', /INFORMATIONAL
  currentPtr=startPtr
  while (PTR_VALID(currentPtr)) do begin

```

```

previousPtr=currentPtr
currentPtr=(*currentPtr).next
endwhile
(*previousPtr).next = PTR_NEW(Setup_llist(bugmsg))
currentPtr=(*previousPtr).next
endif
(*currentPtr).nodePtr = value
(*currentPtr).next = PTR_NEW()

if(bugmsg eq 1 ) then begin
  if NOT(PTR_VALID(value)) then begin
    message,'value is NULL', /INFORMATIONAL
  endif else begin
    message,'value is Valid', /INFORMATIONAL
  endif
  if(NOT(PTR_VALID(startPtr)) or NOT(PTR_VALID(currentPtr)))then begin
    print,'PTR_VALID(startPtr)=',PTR_VALID(startPtr)
    print,'PTR_VALID(nextPtr)=',PTR_VALID(nextPtr)
    print,'PTR_VALID(currentPtr)=',PTR_VALID(currentPtr)
  endif
endif

```

```

if ( bugmsg eq 1 ) then message,'Exited', /INFORMATIONAL

```

```

RETURN, currentPtr

```

END

```

. *****
;
;+
; PURPOSE:
; Return the pointer to the desired record in the linked list.
;
; CALLING SEQUENCE:
;
; EXAMPLE:
;
; REQUIRED ROUTINES:
;
; VARIABLES:
; bugmsg      - Debugging message activator (0-No messages, 1-verbose mode)
;
; MODIFICATION HISTORY:
;   Created: Douglas J. Larson, 05 August, 1998.
;
; -
; =====
=====

```

```

function llist_lookup, bugmsg, startPtr, key, criteria

```

```

if(bugmsg eq 1) then message,' Entered', /INFORMATIONAL

if NOT(PTR_VALID(startPtr)) then begin
  if ( bugmsg eq 1 ) then message,'startPtr is NULL', /INFORMATIONAL
  retval = PTR_NEW()
endif else if (PTR_VALID(startPtr)) then begin
  if ( bugmsg eq 1 ) then message,'startPtr is Valid', /INFORMATIONAL

; Create a pointer to the heap variable pointed at by _startPtr_.
currentPtr = startPtr

    uckey = STRUPCASE(key)

    tag_number = where(tag_names((*currentPtr).nodeptr)) eq uckey, count)
if(bugmsg eq 1) then begin
    print,'counter=',count,' uckey=',uckey,' tag_number=',tag_number
endif

done = 0
while (done eq 0) do begin
  if(bugmsg eq 1) then begin
text='Tag String Requested='+key
message, text, /INFORMATIONAL
help,criteria
;help,(*currentPtr).(tag_number[0])
endif
if(criteria eq (*currentPtr).nodeptr).(tag_number[0])) then begin
  retval = (*currentPtr).nodeptr
  done = 1
endif else begin
  ; Move to the next link in the list
  currentPtr = (*currentPtr).next
  if(ptr_valid(currentPtr) eq 0) then begin
    retval = ptr_new()
    done = 0
  endif
endif
endwhile
endif

if(bugmsg eq 1) then message,' Exited', /INFORMATIONAL

RETURN, retval
END
, *****
;+
; llist_print.pro

```

```

;
; PURPOSE:
;   Print a specified node element of an entire generic linked list.
;
; CATEGORY:
;   Generic singly linked list.
;
; CALLING SEQUENCE:
;   llist_print, bugmsg, startPtr, fieldwanted
;
; KEYWORD PARAMETERS:
;
; COMMON BLOCKS:
;   None
;
; OUTPUTS:
;   Data structure
;
; MODIFICATION HISTORY:
;   Created by: Douglas J. Larson, 22 August, 1998.
;-
; =====
=====
FUNCTION llist_print, bugmsg, startPtr, fieldwanted

if ( bugmsg eq 1 ) then message, ' Entered', /INFORMATIONAL

retval = 1
if NOT(PTR_VALID(startPtr)) then begin
  if ( bugmsg eq 1 ) then message, 'startPtr is NULL', /INFORMATIONAL
  retval = 0
endif else if (PTR_VALID(startPtr)) then begin
  ; Create a second pointer to the heap variable pointed at by _startPtr_.
  currentPtr = startPtr

  ; IDL likes all structure tags in uppercase.
  fieldwanted = strupcase(fieldwanted)

  tag_number = where(tag_names((*currentPtr).nodeptr)) eq fieldwanted, count)

  if(count eq 0)then begin
    message, 'Tag is NOT in structure, check the code!'
    retval = 0
  endif else begin
    ; Traverse the linked list and print the list element information.
    while PTR_VALID(currentPtr) do begin
      print, currentPtr, ',named ', $
        ((*currentPtr).nodeptr).(tag_number[0]), $

```

```
        ', points to: ', (*currentPtr).next
        currentPtr = (*currentPtr).next
    endwhile
endelse
; PTR_FREE,currentPtr
endif

if ( bugmsg eq 1 ) then message,'Exited', /INFORMATIONAL

return, retval
end
```

File Attachments

1) [l1ist.txt](#), downloaded 134 times

Subject: Re: linked list example
Posted by [Mark Hadfield](#) on Thu, 03 Sep 1998 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

A couple of comments:

A linked list just cries out to be implemented as an object class. I wrote such a thing some time ago but will not post the code here for fear of ridicule.

Robert Mallozzi (Robert.Mallozzi@msfc.nasa.gov) has written some container classes, including a last-in-first-out stack and a first-in-first-out queue. The queue is useful when reading data from text files, to avoid the performance hit when extending arrays, as discussed here a week or two ago.

--

Mark Hadfield, m.hadfield@niwa.cri.nz <http://www.niwa.cri.nz/~hadfield/>
National Institute for Water and Atmospheric Research
PO Box 14-901, Wellington, New Zealand

Subject: Re: linked list example
Posted by [davidf](#) on Thu, 03 Sep 1998 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Doug Larson (djl@REMOVECAPSloki.srl.caltech.edu) writes:

> My system administrator would not buy the "Fanning" book
> for me due to tight resources, so I had to invent my
> own (kluge) solutions.

I'm sending Mr. Larson a book with my compliments for adding to the code resources of this newsgroup. It's actually the last one left of the original first edition. The updated second edition hits the streets tomorrow. Many thanks to all the generous people on this newsgroup who have bought books and offered suggestions and corrections. The next book will be coming "real soon now". :-)

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438, Toll-Free Book Orders: 1-888-461-0155
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>

Subject: Re: linked list example
Posted by [mirko_vukovic](#) on Fri, 04 Sep 1998 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

In article <6sl5km\$m7d\$1@clam.niwa.cri.nz>,
"Mark Hadfield" <m.hadfield@niwa.cri.nz> wrote:
> A couple of comments:
>
> A linked list just cries out to be implemented as an object class. I wrote
> such a thing some time ago but will not post the code here for fear of
> ridicule.
>
actually, IDL_Container is a linked list. But it accepts only objects as its
elements (last time I looked, months ago)

Mirko

-----== Posted via Deja News, The Leader in Internet Discussion ==-----
http://www.dejanews.com/rg_mkgrp.xp Create Your Own Free Member Forum

Subject: Re: linked list example
Posted by [davidf](#) on Fri, 04 Sep 1998 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Mark Hadfield (m.hadfield@niwa.cri.nz) writes:

- > A linked list just cries out to be implemented as an object class. I wrote
- > such a thing some time ago but will not post the code here for fear of
- > ridicule.

I routinely post code here when I don't have the time or energy to fix it up myself. Someone always grabs it and a day later e-mails me the "corrected" code with all those features I should have added in the first place.

If I'm just searching for bugs, I announce that the program "works" or is "the best program I've ever written".

I've been thinking of myself lately as more of an "idea person" rather than a "programmer". It's easier on the ego. :-)

- > Robert Mallozzi (Robert.Mallozzi@msfc.nasa.gov) has written some container
- > classes, including a last-in-first-out stack and a first-in-first-out queue.
- > The queue is useful when reading data from text files, to avoid the
- > performance hit when extending arrays, as discussed here a week or two ago.

You know, even with as much free code as there is out there, I find that not too many people use it. I don't really know why. Perhaps it is just hard to read other people's code, or maybe you don't really understand it unless you write your own (my reason for writing a linked list rather than using the one Beau Legeer uses in the Advanced Programming classes). In any case, I found I had to write a book to call any attention at all to the programs I offer.

Now that I think about it, maybe I should revisit the notion of program quality. :-)

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438, Toll-Free Book Orders: 1-888-461-0155
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>
