

Forum:[Hack Forums](#)

Thread: [C - Linked List with void* instead of node*](#)

theChive - October 4, 2017, 11:38 am

Hey all, I'm stumped on a school assignment. I'm not looking for you to do my work for me, just a little nudge forward.

I have this struct that I need to create a linked list with and reverse it. Easy. However, the catch is that instead of having a pointer next to the next struct in the list, I have a void pointer that (required by my professor) points to the corresponding void pointer in the next struct in the list. I'm not sure how to even work this out.

Anyone have some suggestions for just getting my mind around this and getting started? Thanks!

Here's my header file:

Code struct _Node {

union {

int n;

char c;

} val;

void *ptr;

int var;

};

typedef struct _Node Node;

Edit: The header file is given and can't be changed by me, by the way.

?lpha - October 4, 2017, 11:45 am

(Oct 4, 2017 11:38 AM)theChive Wrote: [\[CITE\]](#) Hey all, I'm stumped on a school assignment. I'm not looking for you to do my work for me, just a little nudge forward.

I have this struct that I need to create a linked list with and reverse it. Easy. However, the catch is that instead of having a pointer next to the next struct in the list, I have a void pointer that (required by my professor) points to the corresponding void pointer in the next struct in the list. I'm not sure how to even work this out.

Anyone have some suggestions for just getting my mind around this and getting started? Thanks!

Here's my header file:

Code struct _Node {

union {

int n;

char c;

} val;

void *ptr;

int var;

};

typedef struct _Node Node;

Edit: The header file is given and can't be changed by me, by the way. An address is just an address, it doesn't have a type so it's just void. When you want to dereference it, you should typecast it to a

struct _Node. So that the compiler knows how many bytes it should reserve.

theChive - October 4, 2017, 11:58 am

(Oct 4, 2017 11:45 AM)?lpha Wrote: [\[CITE\]](#) An address is just an adress, it doesn't have a type so it's just void. When you want to dereference it, you should typecast it to a struct _Node. So that the compiler knows how many bytes it should reserve.

Ahh ok. Thanks!

Any idea if the location of the void pointer within the struct will make a difference? Like if I'm reversing the list, will I have to offset the address by some factor since the void pointer points to the next struct's void pointer, not the next struct?

I got it! Thanks for you help pointing me in the right direction, Alpha!!