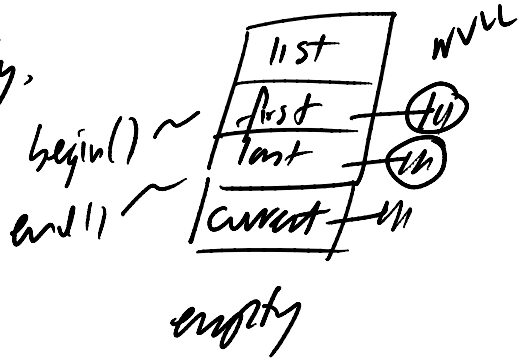
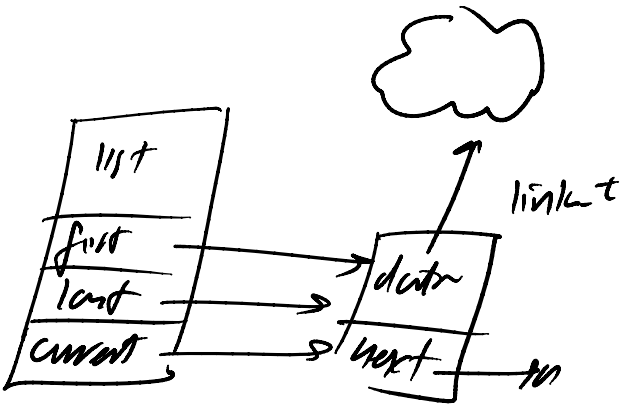
Wednesday, November 17, 2010
8:51 AM

Previously,



empty

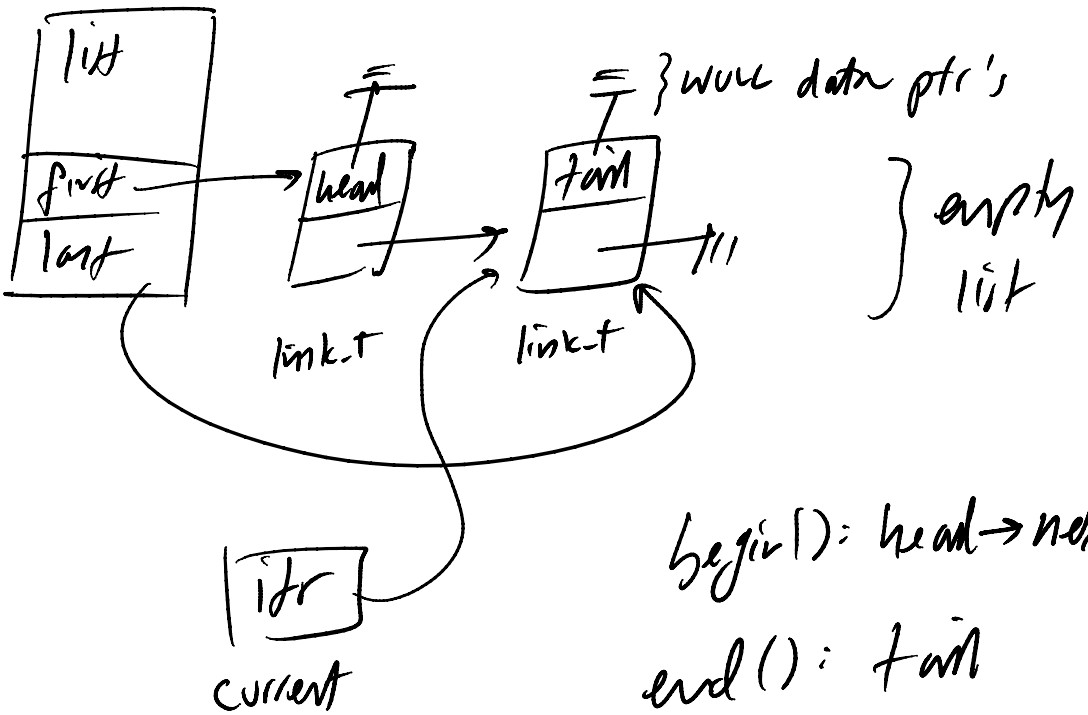


with one node on list.

Problem: with one node on list, can't iterate via

```
for(itr = list.begin(); itr != list.end(); ++itr)
```

V1.0



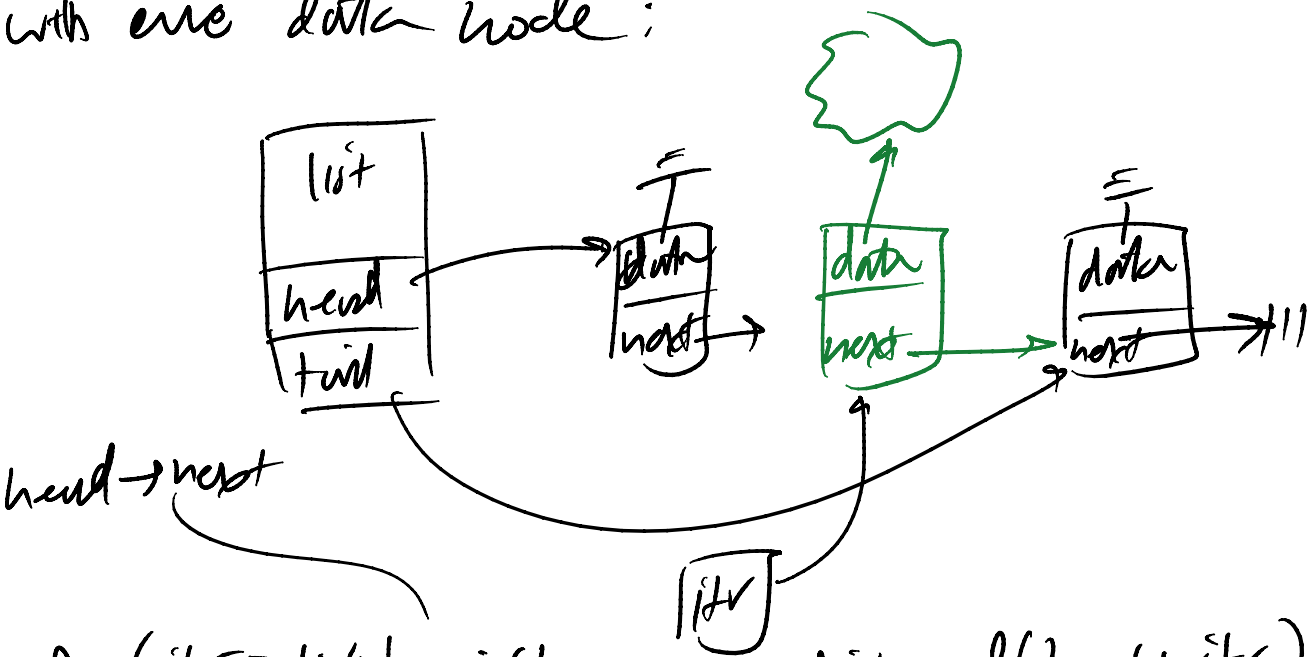
begin(): head->next

end(): tail

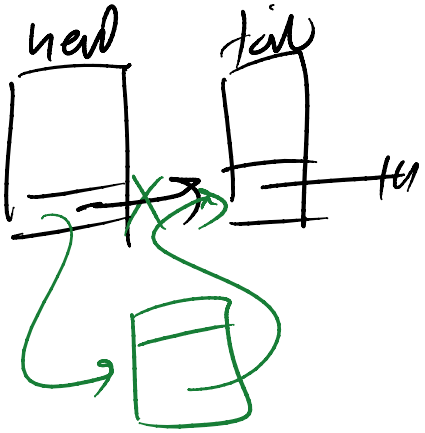
for(itr = list.begin(); itr != list.end(); ++itr)

works if list empty because begin() == end()
head->next tail

with one data node:

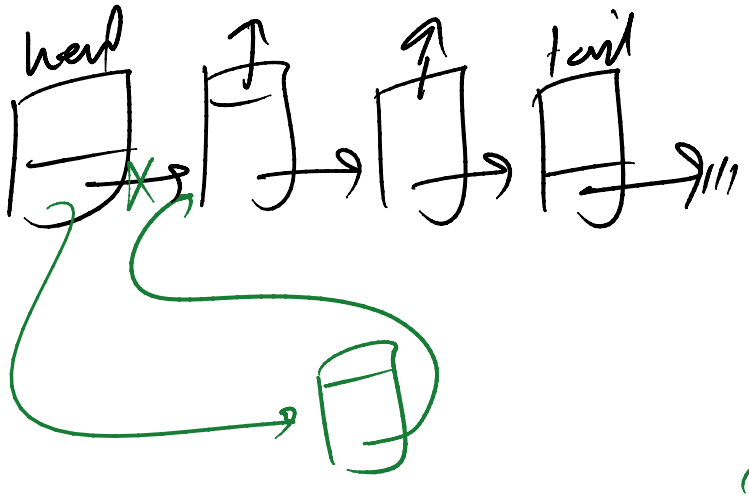


for(itr = list.begin(); itr != list.end(); ++itr)
std::cout << *itr;



1. tail->next = head->next

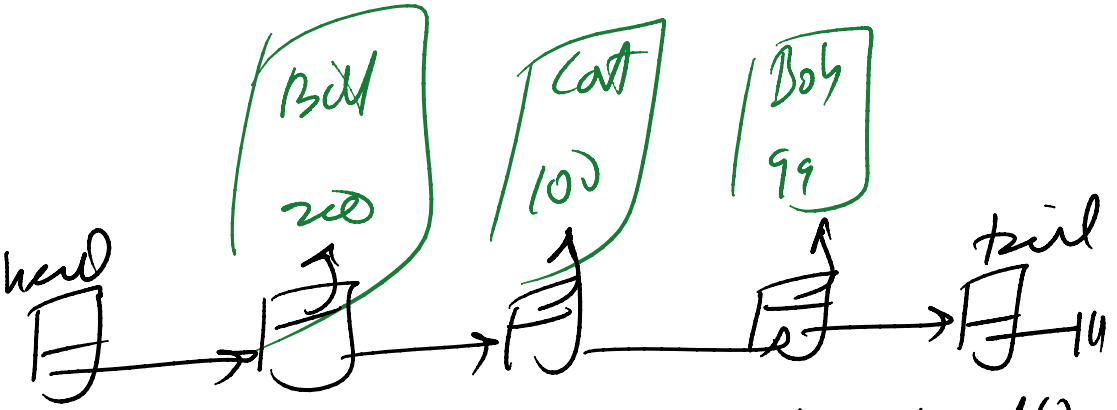
head->next = link



Bob 99

Cat 100

Bill 200




```
for (itr = list.begin(); itr != list.end(); ++itr)
    std::cout << *itr;
```

Bill 200

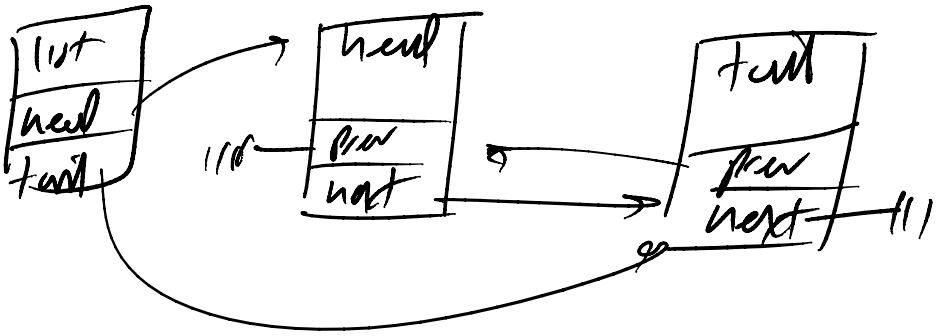
Cat 100

Bob 99

Problem: everything works
but insertion is at front, hence
list printout is in reverse order

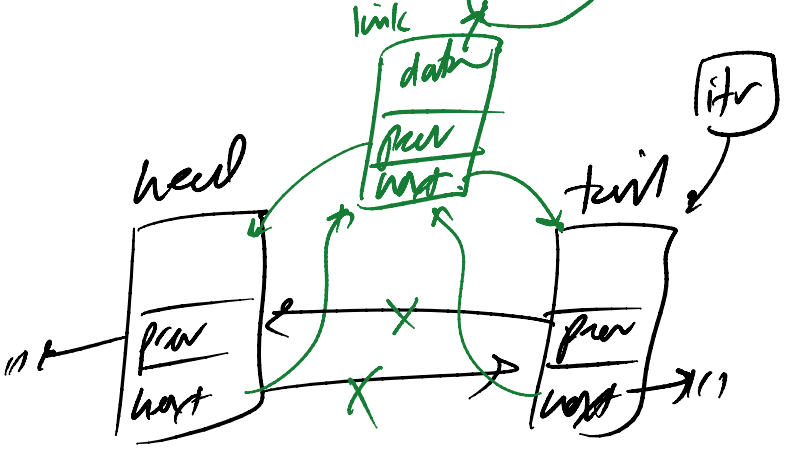
Soln: doubly-linked list

V2.0



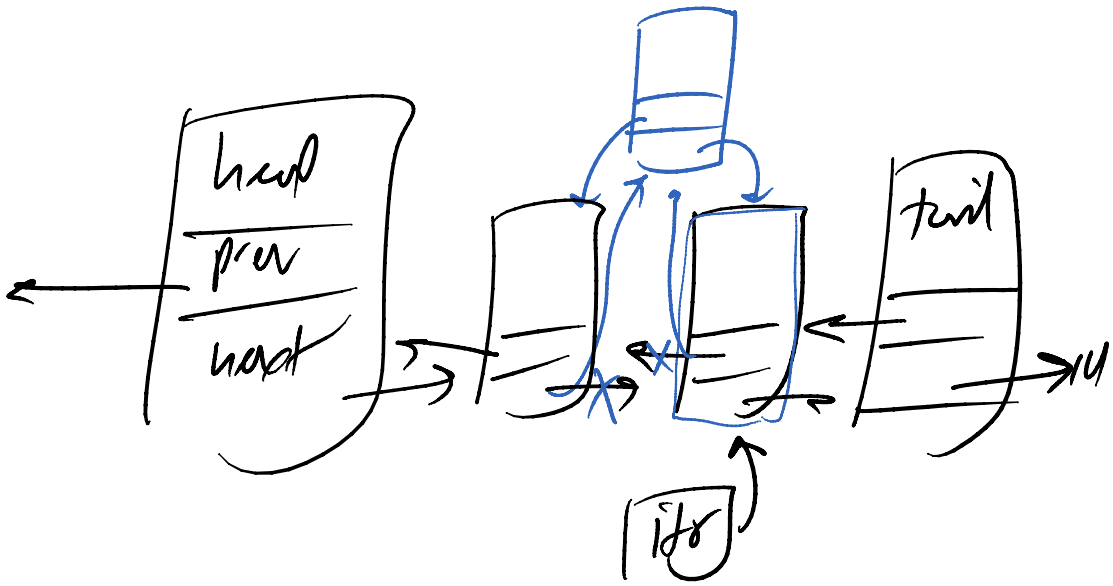
① redesign link_t structure
link_t now has two pointers, prev, next

② insertion



link->prev = itr->prev

link->next = itr->prev->next;



itr->prev->next = link

itr->prev = link

- insertion at end of list ⇒ list printout works

- head/tail nodes ⇒ list iterator works even if list is empty, has one or more items

- summary:

- link_t structure has prev, next ptrs
- list_t has head, tail ptrs.
  (2 "dummy" link_t nodes)
- list_t has iterator to replace "current" ptr.