Linked List Operations

15-123

Systems Skills in C and Unix

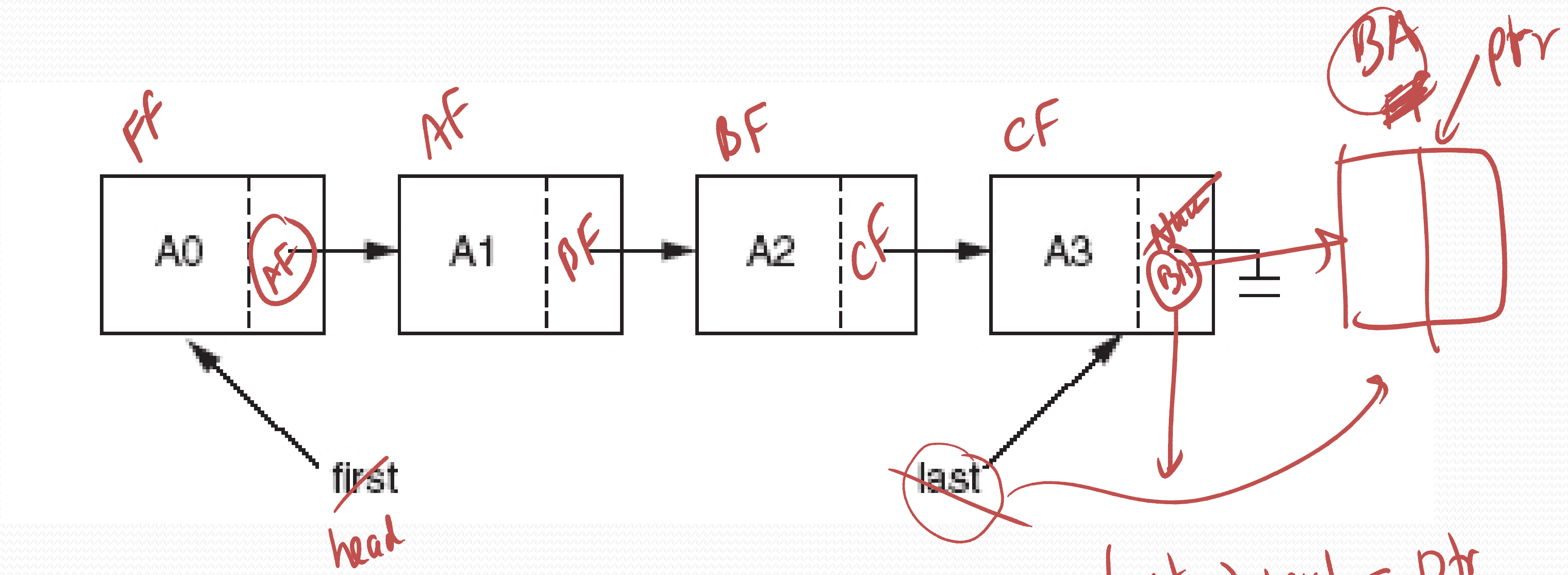
Why Linked Lists?

- Flexible memory management
- Easy adds and deletes from a list $O(1)$
- A data structure you would always consider using

When size of the list is unknown

Types of Linked Lists

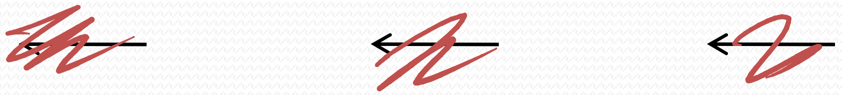
Singly Linked Lists

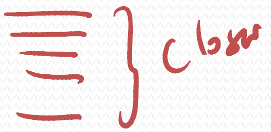


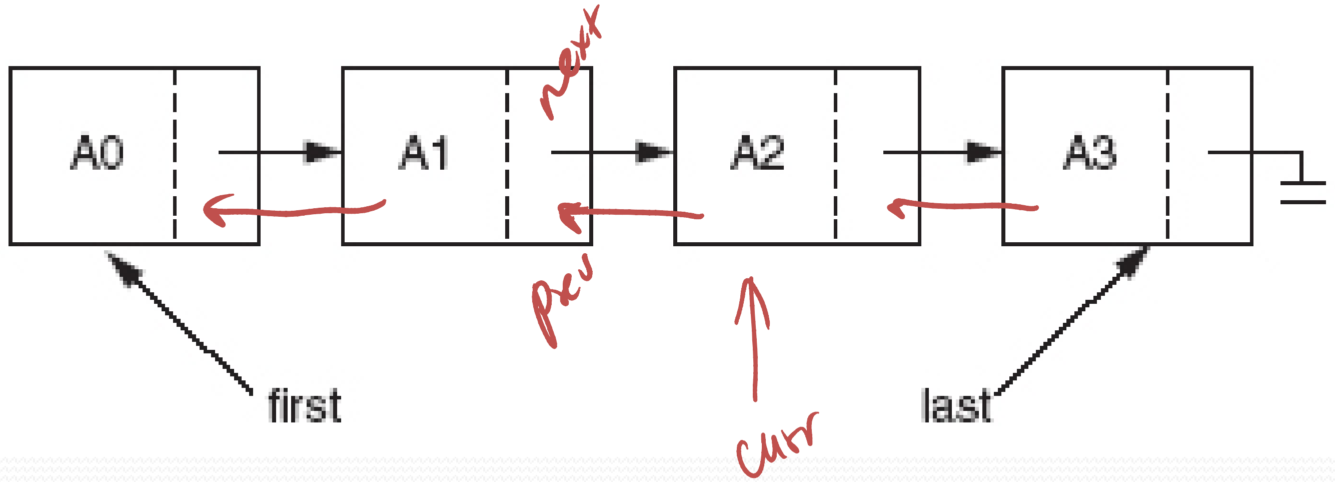
last → next = ptr

last = ptr

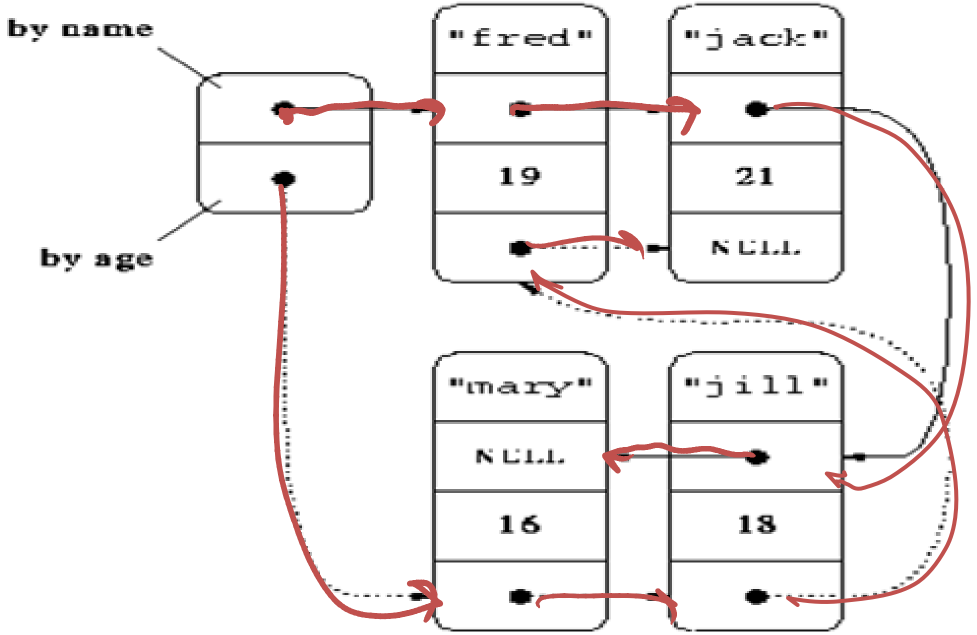
Doubly Linked Lists





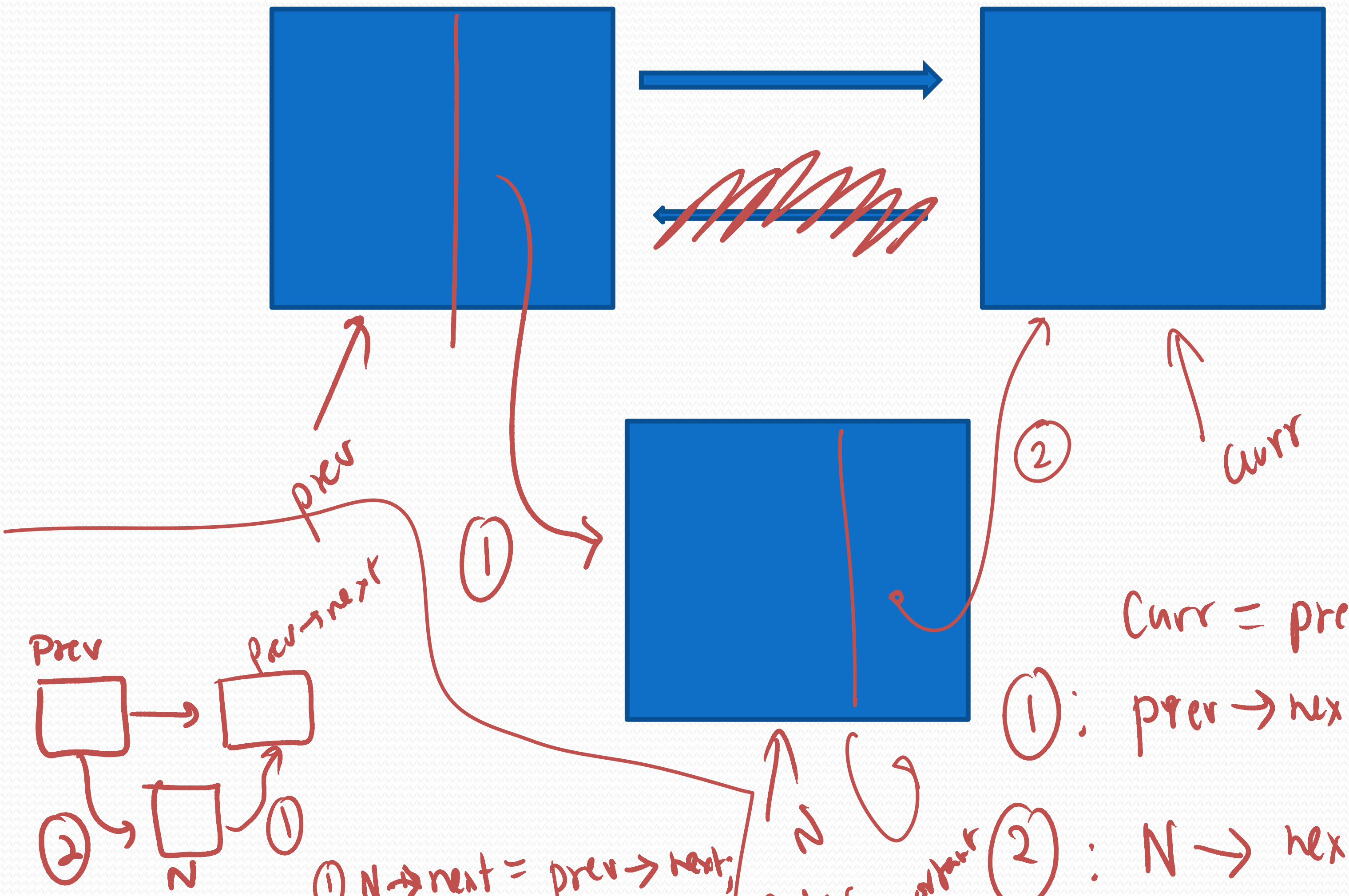


Multilinked Lists



Linked List operations on DLL’s

Adding Nodes (LL)



Curr = prev → next;

① : prer → next = N

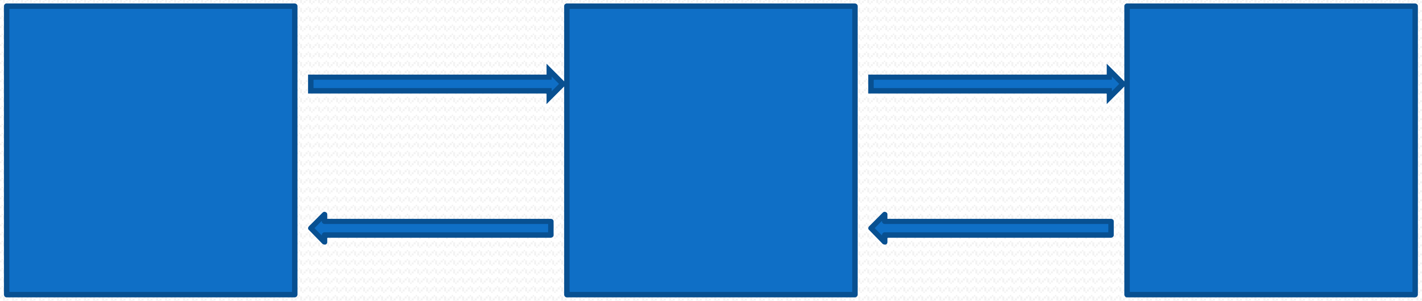
② : N → next = Curr

① N → next = prev → next;

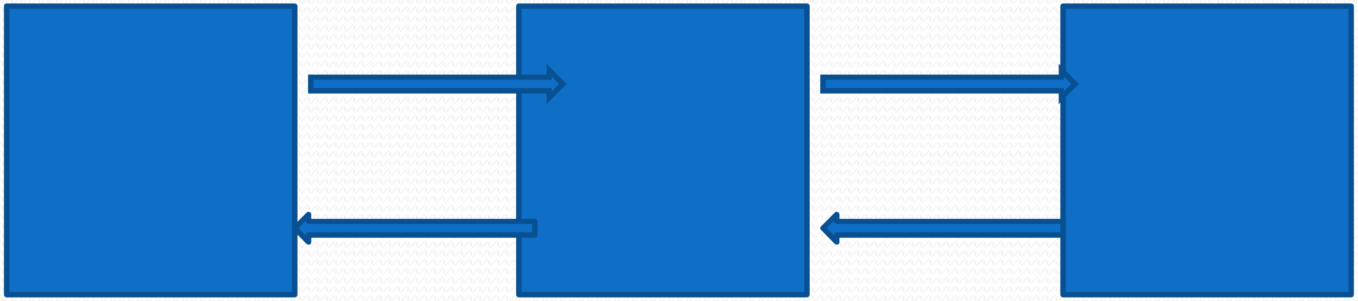
② prrv → next = N;

Order is important

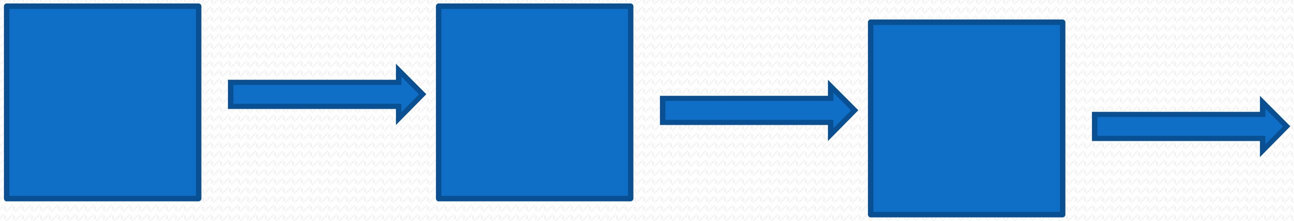
Deleting Nodes



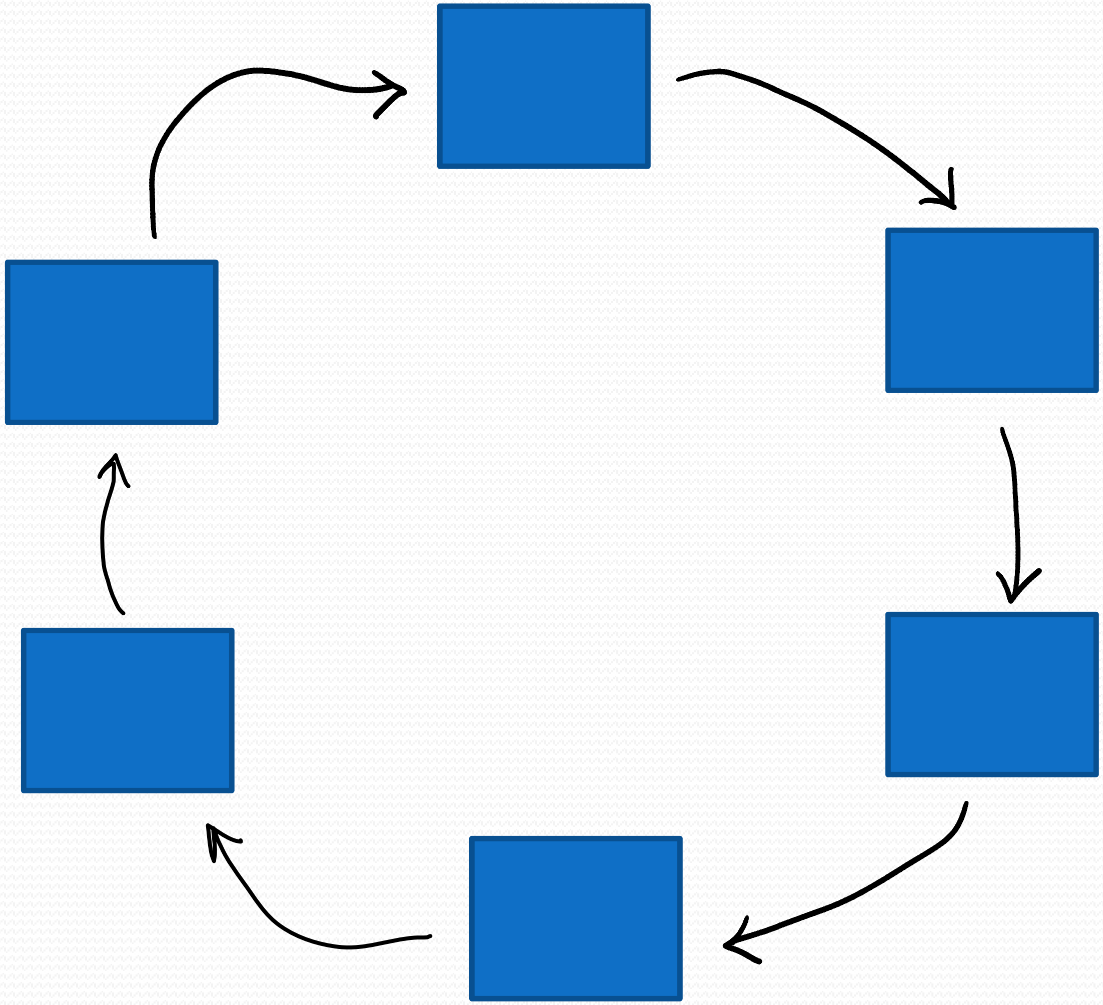
Traversing a DLL



Making a singly LL circular



Rotating a circular LL



Things we should know about LL’s

- Understand the difference between a LL node and a pointer to a node
- Head is typically NOT a node, but a pointer to the first node
- Be careful dealing with LL nodes, as misguided link could create infinite loops, memory leaks or incorrect outputs

Coding Examples