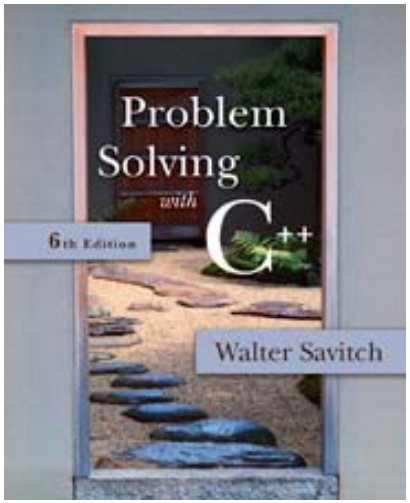
APS105: Lecture 34

Wael Aboelsaadat

wael@cs.toronto.edu

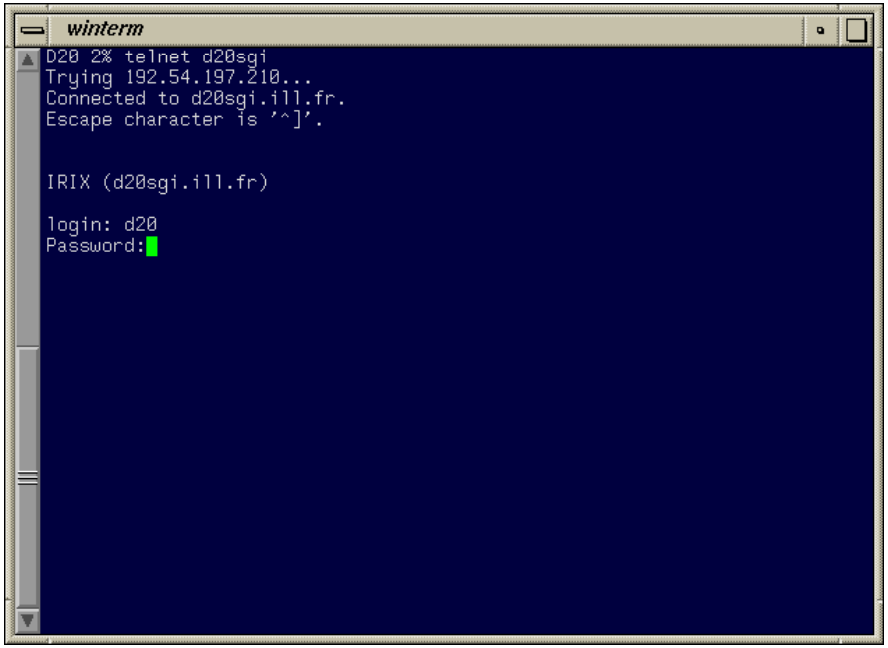
http://ccnet3.utoronto.ca/20079/aps105h1f/

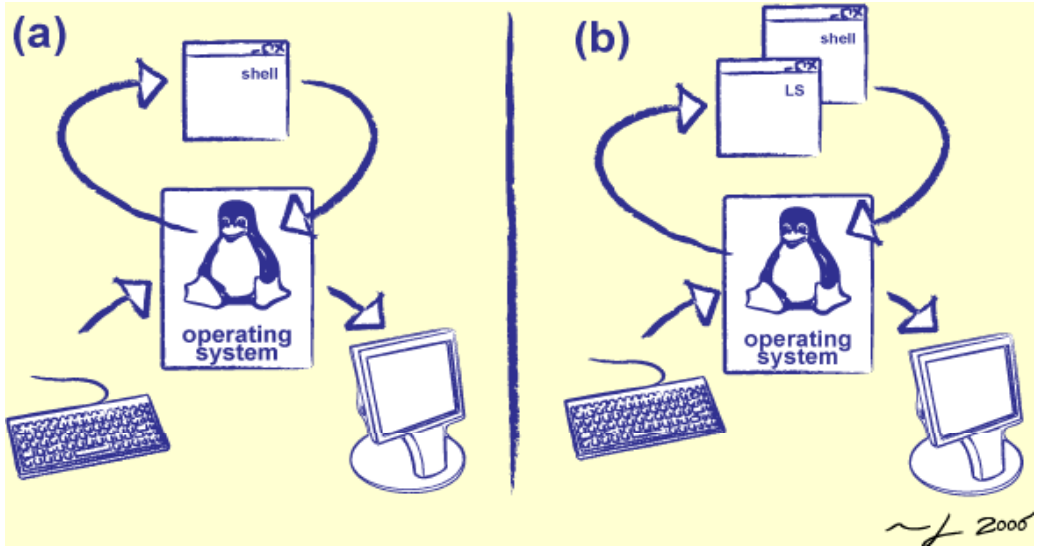
Acknowledgement: These slides are a modified version of the text book slides as supplied by Addison Wesley



What is a Shell?

- ... a UNIX shell is a program that accepts and interprets commands and then has the operating system execute them.





- It is *not* the operating system

Shell: Input/Output Redirection

- Most programs have three I/O streams:
  - *stdin* – standard input
  - *stdout* – standard output
  - *stderr* – standard error.
- They all default to the console ("console" means the keyboard for the input and the screen for the output)

Shell: Input/Output Redirection

- To redirect *stdout* of a program to a file:

  bash: `myprogram 1> output.log`

  tcsh: `myprogram > output.log`

- To redirect *stderr* of a program to a file:

  bash: `myprogram 2> error.log`

- To redirect both *stdout* and *stderr* to same file (order matters):

  bash: `myprogram > combined.log 2>&1`

- To redirect both *stdout* and *stderr* separately:

  `(myprogram >output.log) >&error.log`

Shell: Input/Output Redirection

- Example of “>>” operator to append information to a file:

```
prompt> date > foo

prompt> cat foo
Wed Aug 31 17:27:52 CDT 2005

prompt> date >> foo

prompt> cat foo
Wed Aug 31 17:27:52 CDT 2005
Wed Aug 31 17:27:56 CDT 2005
```

Shell: Input/Output Redirection

function	tcsh	bash
stdout to file	comm >ofile	comm > file
stderr to file		comm 2> file
stdout/err to file	comm >& ofile	comm >ofile 2>&1 comm &> ofile
stdin from file	comm < ifile	comm < ifile
stdout to end of file	comm >> ofile	comm >> ofile
stderr to end of file		comm 2>> ofile
stdout/err to end	comm >>& ofile	comm >> ofile 2>&1
stdin until “c”	comm <<c	comm <<c
redirect stdin/out	comm < ifile > file	comm < ifile > file
stderr to third file		comm < ifile > ofile 2> efile
stdout/err to 2 files	(comm > ofile) >& efile	

I/O Redirection - Pipes/Filters

- Pipelines are a set of processes chained by their standard streams, so that the *stdout* of each process feeds directly as the *stdin* of the next.
- Pipelines are defined using the "|" character.

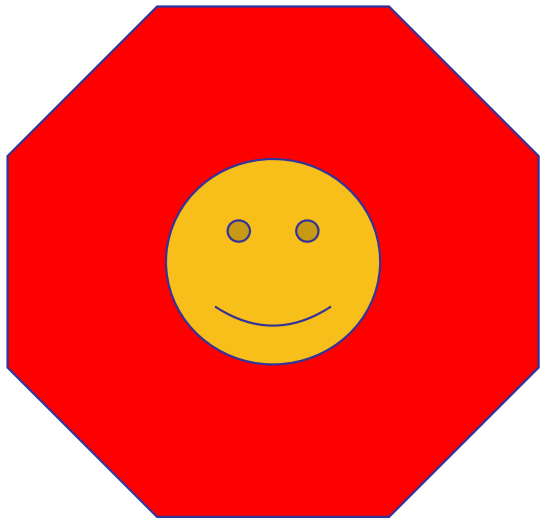
function	tcsh	bash
pipe stdout to comm2	comm | comm2	comm | comm2
pipe stdout/err to comm2	comm |& comm2	comm 2>&1 | comm2

- E. g.,

```
ls | more

ls –la | more

ls –la | more | print
```



The End!

Revision: pointers & recursion

- Tree Data structure

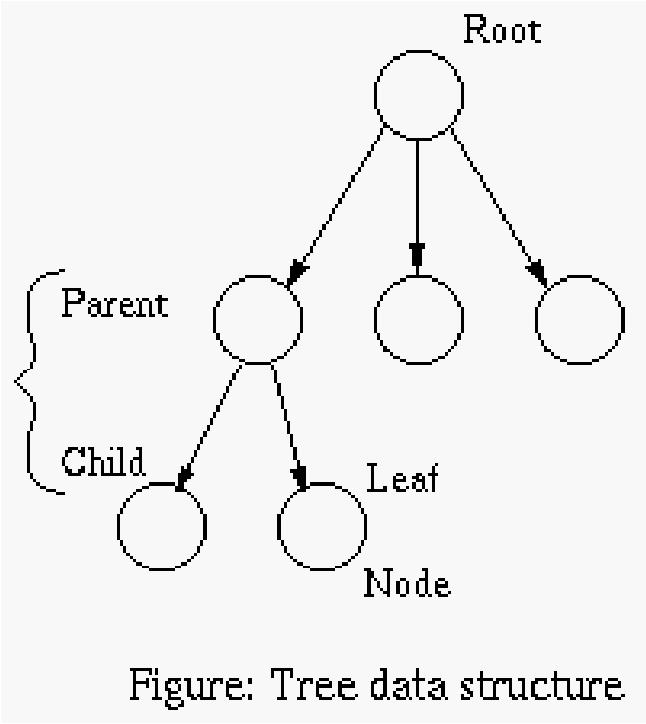
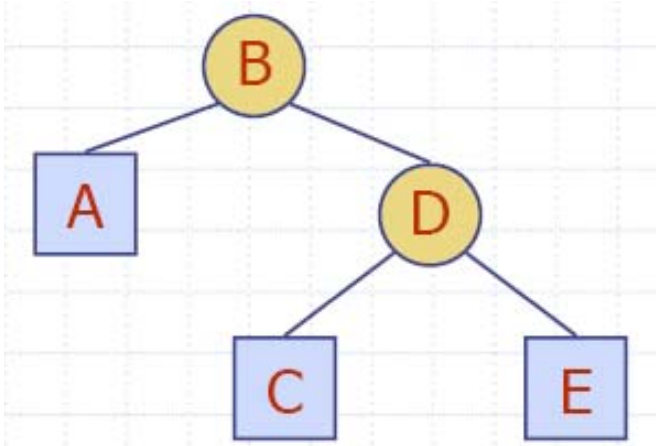
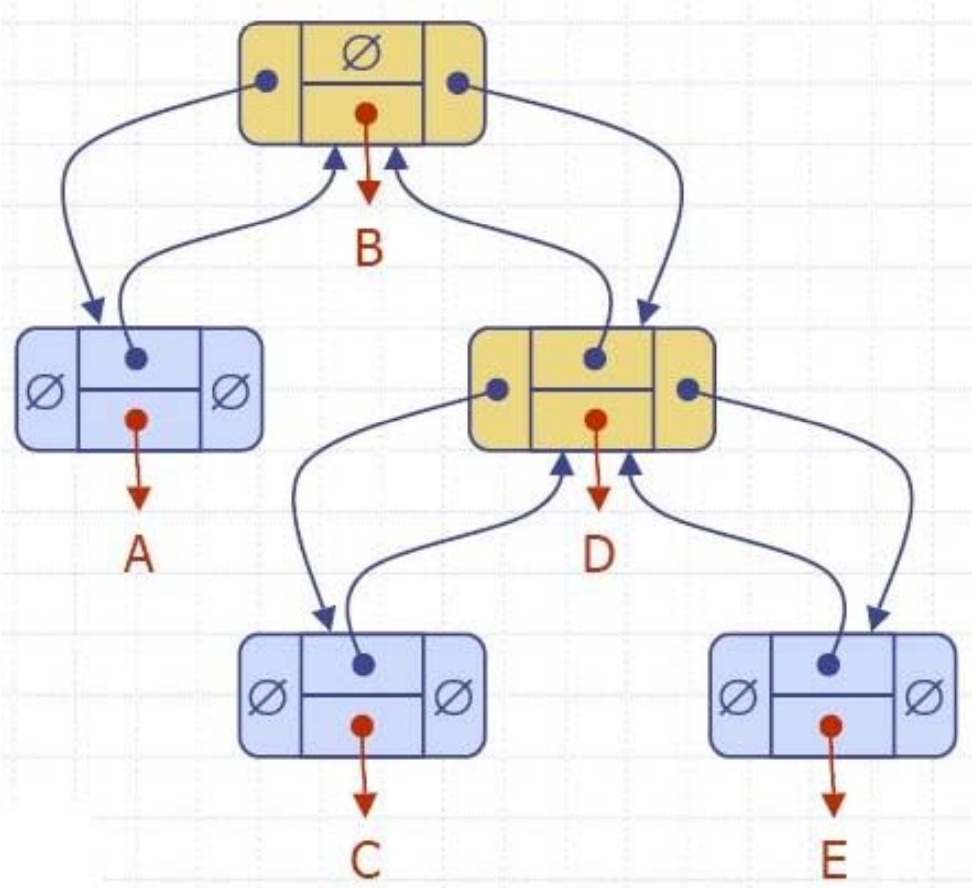


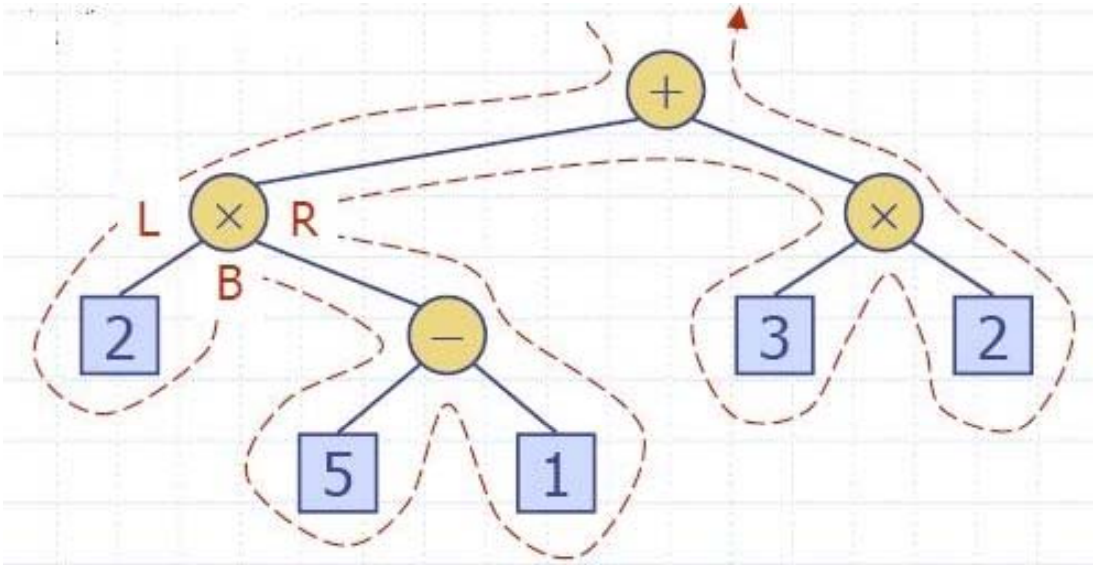
Figure: Tree data structure

Binary Trees





Binary Tree Traversal



http://www.cosc.canterbury.ac.nz/mukundan/dsal/BTree.html
http://en.wikipedia.org/wiki/Tree_traversal

Binary Tree Traversal

```
preorder(node)
  print node.value
  if node.left  ≠ null then preorder(node.left)
  if node.right ≠ null then preorder(node.right)
```

```
inorder(node)
  if node.left  ≠ null then inorder(node.left)
  print node.value
  if node.right ≠ null then inorder(node.right)
```

```
postorder(node)
  if node.left  ≠ null then postorder(node.left)
  if node.right ≠ null then postorder(node.right)
  print node.value
```

Binary Tree Search (pseudo code)

```
NodePtr searchBinaryTree(const NodePtr T, NodePtr &v, int nValue)
{
   if (v.value == nValue)
          return v;              // found!
   else
          if (v.value < nValue )
                 searchBinaryTree(T, T.ptrLeft, nValue);   // search left subtree
          else
                 searchBinaryTree(T, T.ptrRight, nValue); // search right subtree

}
```