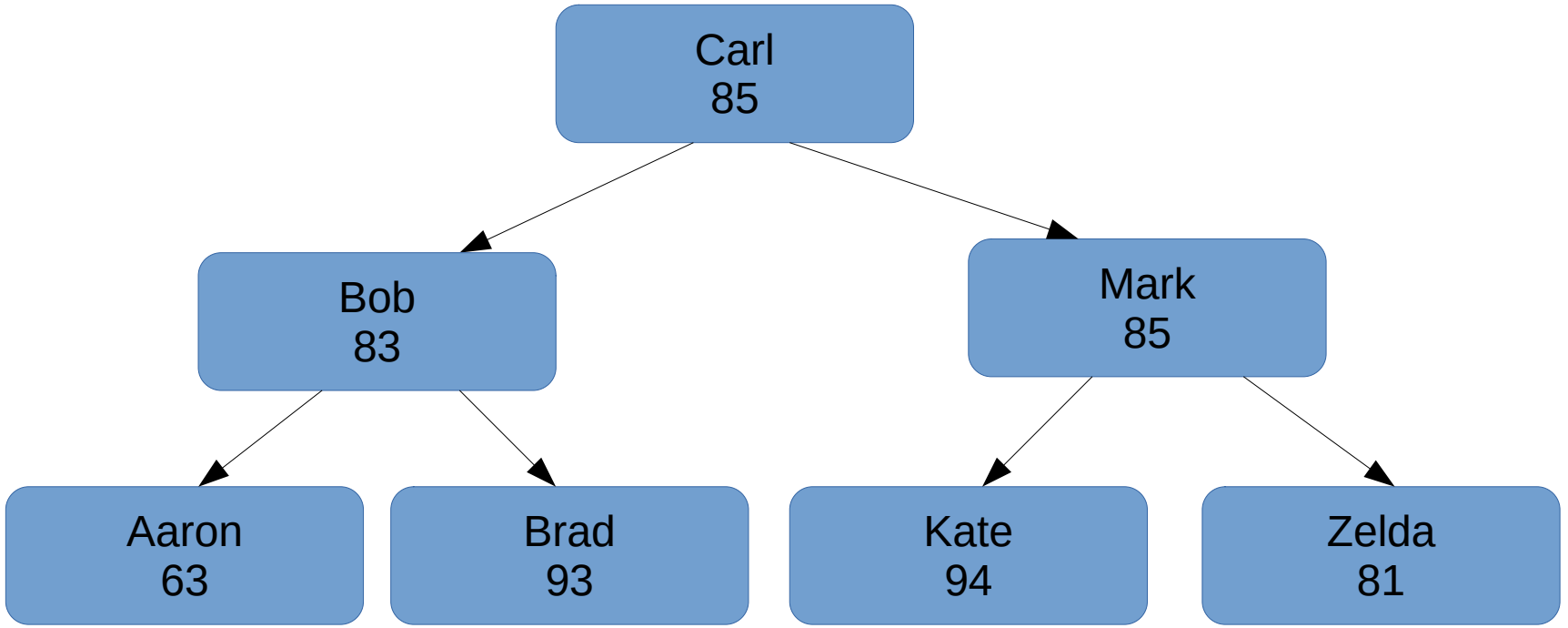
Software Tools

C, Unix (Linux), and tools

Binary Tree Example



Defining BinTree Structure

- A very basic binary tree (should be in a .h file)

```
typedef struct bintreestruct *bintree;
struct bintreestruct {
  int num;
  char *s;
  bintree l, r;
};
```

Functions for Binary trees

- Also in a .h file.

```
extern bintree mk_bintree(void);
extern void free_bintree(bintree node);
extern void insbintree(int i, char *s, bintree *t);
extern char *srchbintree(int i, bintree t);
extern void printbintree(bintree t);
extern int heightbintree(bintree t);
```

Allocating a node

- Should be in a .c file.

```
bintree mk_bintree(void)
{
  bintree res;

  res = (bintree) malloc(sizeof(struct bintreestruct));
  if (res==NULL)
    {
      fprintf(stderr,"Fatal Error: not enough memory\n");
      exit(1);
    }
  res->l = NULL;
  res->r = NULL;
  res->s = NULL;
  return res;
}
```

Deallocating a tree

- Should be in a .c file.

```
void free_bintree(bintree node)
{
  if (node!=NULL)
    {
      free_bintree(node->l);
      free_bintree(node->r);
      if (node->s != NULL) free(node->s);
      free(node);
    }

}
```

Printing a tree

- Should go in a .c file

```
void printbintree(bintree t)
{
  if (t == NULL) return;
  printbintree(t->l);
  printf("%6d: %s\n",t->num, t->s);
  printbintree(t->r);
}
```

Height of a Tree

- As always in a .c file

```
int heightbintree(bintree t)
{
  int hl, hr;
  if (t==NULL) return 0;
  hl = heightbintree(t->l);
  hr = heightbintree(t->r);
  if (hl>hr)
    return hl+1;
  else
    return hr+1;
}
```

Standard I/O Library

- Contains many functions that help us with I/O
- All function declarations are in stdio.h which has to be included.
- These are functions that invoke system calls that do the actual I/O but are easier to use.
- These functions are meant to be portable, unlike the system calls that depend on the OS.

Getchar and Putchar

- Perform I/O character by character.

```
while ((c=getchar()) != EOF)
  putchar(isupper(c) ? tolower(c) : c);
```

Formated output

- Printf *et al*.
- Uses a formating string with formating directives
- Directives start with a % and end with a conversion character.
- After the % there might be a – (left alignment), a minimum field width, a . followed by the precision and a length modifier (l, letter ell)

Conversion Characters

```
d: decimal
o: octal
x: hexadecimal
u: unsigned
c: character
s: string
E, e: real number with exponential notation
f: decimal notation for reals (floats or doubles)
g: either %e or %f
```