Software Tools

C, Unix (Linux), and tools

Portability Issues

- This works fine for ASCII
- Does not work for EBCDIC
- So just in case you ever encounter an EBCDIC computer you should use:
  - isdigit(c)
  - Instead of
  - c>=’0’ && c<=’9’
- C has several functions that handle these subtle issues.

Type promotion

- When we add (subtract, whatever) two entities of different type the “lower” type is “promoted” to the “higher”
- `Long double` is higher than `double`, higher than `float`, `int`.
- For integral types things get a bit tricky:
  - If an integral type can be converted to an appropriately sized `int` without loss it is. O/w it is converted to an appropriately sized `unsigned int`.
  - Integer conversions happen by truncation, zero padding or sign extension. Often are implementation dependant.
  - Thus `-1L>1UL` !!!

Pseudo-random function.

```
unsigned long next=1;
int rand(void)
{
  next = next * 1103515245 + 12345;
  return (unsigned)(next/65536)%32768;
}
```

Increment-Decrement Operators

- Very useful operator. Makes it easy to write compact code
- Comes in two flavours: prefix and postfix.
  - The value of `n++` (postfix) is the original value of `n`.
  - The value of `++n` (prefix) is the new value of `n`.
- Similarly for decrementing `--n` and `n--`

Increment-Decrement Operators

- Very useful operator. Makes it easy to write compact code
- Comes in two flavours: prefix and postfix.
  - The value of `n++` (postfix) is the original value of `n`.
  - The value of `++n` (prefix) is the new value of `n`.
- Similarly for decrementing `--n` and `n--`

getline

```
int KRgetline(char s[], int lim)
{
  int c, i;

  for (i=0; i<lim-1 && (c=getchar())!=EOF && c!='\n'; i++)
    s[i] = c;
  if (c=='\n') {
    s[i] = '\n';
    i++;
  }
  s[i] = '\0';
  return i;
}
```

String concatenate (strcat)

```
int KRstrcat(char s[], char t[])
{
  int i, j;

  i=j=0;
  while (s[i]!='\0') i++;
  while ( (s[i++]=t[j++]) != '\0');
}
```

Bitwise operators

- Very useful for manipulating bits. This is how it is done in C.
- These are:
  - `&`
  - `|`
  - `^`
  - `<<`
  - `>>`
  - `~`
- They are different from logical operators (`&&` or `||`)

Bitwise operators

- We can get the last 8 bits of an `int`:
  - x = x&0xFF;
  - x &= 0xFF;
- We can get the previous 7 bits:
  - x = (x>>8)&0xFF;
- We can set the previous 7 bits x to the last 7 bits of y:
  - x = (x&(0177<<8)) | ((y&0177)<<8);

Fancy Assignment Operators

- Another great thing about C is the fancy assignment operators like:
  - `i += 2;`
- Most binary operators have their assignment version.
- Very useful when the lhs is a messy little animal:
  - `yyval[parse.current + parse.offset] += 2;`
- Easy to read, easy to write.

Conditional Expressions

- We all know the good old `if` statement.
- There is also the `switch-case` statement.
  - Be careful with this, it is tricky.
- C has also conditional expressions:
  - `Z = (Z>=0)?Z:-Z;`
  - This is the absolute value of Z.
- Can make the code more compact and/or more readable.
- Allow tricky `#define`d macros.

Problems

- Write a function `int invert(int x, int p, int n)` that inverts bits `p...p+n` of `x`.
- Write a function `void ToHighLow(int n, char s[])` that accepts an integer n and a string s that has enough space and writes out the binary version of it but instead or writing zeros and ones it writes `H` (for high or 1) and `L` (for Low or 0). Leading `L`s are omitted.