# Software Tools

C, Unix (Linux), and tools

## The Fahrenheit Program (so far)

```
#include <stdio.h>

int main()
{
  float fahr, celcius;
  int lower, upper, step;

  lower = 0;
  upper = 300;
  step  = 20;

  fahr = lower;
  while (fahr <= upper)
    {
      celcius = 5*(fahr-32)/9;
      printf("F:%5.2f\tC:%5.2f\n", fahr, celcius);
      fahr += step;
    }
}
```

## C Preprocessor

- Before the C compiler does the actual compilation, it does some preprocessing
- This is a macro processor (a textual transformer)
- We have seen a preprocessor directive already: the `#include <stdio.h>`
  - This inserts the file `/usr/include/stdio.h` into our program before compilation
- Next we see the `#define`.

## Defining Constants

- We often need to define constants in our programs
  - Variables `lower`, `upper`, and `step` make little sense as constants.
- This is what `#define` is for.
- We can use it to define a constant and whenever the preprocessor sees an instance of this contant, it replaces it with its value.

## The Fahrenheit Prog. (constants)

```
#include <stdio.h>

#define LOWER 0
#define UPPER 300
#define STEP 20

int main()
{
  float fahr, celcius;

  fahr = LOWER;
  while (fahr <= UPPER)
    {
      celcius = 5*(fahr-32)/9;
      printf("F:%5.2f\tC:%5.2f\n", fahr, celcius);
      fahr += STEP;
    }
}
```

# For Loops

- A very flexible and compact concept
- Easy for humans to read
- Easy for compilers to optimize

## The Fahrenheit Prog. (for loop)

```
#include <stdio.h>

#define LOWER 0
#define UPPER 300
#define STEP 20

int main()
{
  float fahr, celcius;

  for (fahr=LOWER; fahr<=UPPER; fahr+=STEP)
    {
      celcius = 5*(fahr-32)/9;
      printf("F:%5.2f\tC:%5.2f\n", fahr, celcius);
    }
}
```

# Input Copying

- We introduce getchar and putchar.
- Function `getchar` gets a character from the input and returns it as an integer.
- If an *end-of-file* (`EOF`) is encountered, it returns -1 (hence we need the integer)
- We also see some more tricks of the trade
  - Assignment operation returning a value
  - A standard symbolic constant

# The getputchar Prog.

```
#include <stdio.h>

int main()
{
  int c;

  c=getchar();
  while (c!=EOF)
    {
      putchar(c);
      c=getchar();
    }
  printf("Got %d\n",c);
}
```

# Character Counting

- We write a simple program using both a `while` and a `for` loop.
- We play with the `++` operator
- And write an extremely short `for` loop
- Introduce the `long` integer

# The charcnt Prog. (while)

```
#include <stdio.h>

int main()
{
  long nc;

  nc = 0;
  while(getchar()!=EOF)
    ++nc;
  printf("%5ld\n",nc);
}
```

## The charcnt Prog. (for)

```
#include <stdio.h>

int main()
{
  long nc;

  for (nc=0; getchar()!=EOF; nc++)
    ;
  printf("%5ld\n",nc);
}
```

## Word Counting

- Write a program that counts words
- Play with the if statement
- Introduce character constants
- Introduce the `||` operator
- Double check against the Linux command `wc`.

# The wordcnt Prog.

```
#include <stdio.h>
#define INWRD 1
#define OUTWRD 0

int main(){
  long nc, nl, nw;
  int c, state;
  state = OUTWRD;
  nc = nw = nl = 0;
  for ( ; (c=getchar())!=EOF; nc++){
      if (c=='\n')  nl++;
      if (c==' ' || c=='\n' || c=='\t')
        state = OUTWRD;
      else if (state==OUTWRD){
        state = INWRD;
      nw++;
      }
  }
  printf("%5ld %5ld %5ld\n", nl, nw, nc);
}
```

## Arrays

- An easy intro to arrays
- Arrays in C are really pointers
  - But do not worry about it for now.
- We see how we define fixed size arrays
- We see how we assign values to them

## The wordcnt Prog.

```
#include <stdio.h>

int main()
{
   int c, i;
   int ndigit[10];

   for (i=0; i<10; i++)  * Set all elements to zero *
      ndigit[i]=0;

   while ( (c=getchar()) != EOF )
      if ('0'<=c && c<='9')
         ndigit[c-'0']++;
   printf("# of digits: ");
   for (i=0; i<10; i++)
      printf(" %d", ndigit[i]);
   printf("\n");
}
```

## Functions

- See how we declare functions
- How we define functions
- How we return values
- How we specify the parameters (the variables that appear in the function header)
- How we pass arguments (the values that we give to a function when we invoke it)

## The wordcnt Prog.

```
#include <stdio.h>

int power(int m, int n);   /* function declaration */
                          /* aka function prototype */
int main()
{
  /* .... */
  return 0;
}

int power(int base, int n)    /* function definition */
{
  int i, p;

  for (i=1, p=1; i<=n; i++)
    p *= base;
  return p;
}
```