Australian National University

Simple IO

Eric McCreath

putchar

We have looked at printf and it is a simple way of outputting to *standard out* . If you want something even simpler (and a little more control) then putchar gives you a way of outputting a single character at a time.

Below is an example that puts lots of a's to *standard out* .

```
#include<stdio.h>

int main() {
   while (1) putchar('a');
   return 0;
}
```

getchar

'getchar', the partner of 'putchar', can obtain a single character from standard in.

Below is an example of the two working together with putchar echoing anything getchar reads.

```
#include<stdio.h>

int main() {
   while (1) putchar(getchar());
   return 0;
}
```

getchar() returns an EOF integer when it reaches the end of the file. Note files don't really have end characters this is something just created by the getchar function.

scanf

If you are just doing some simple parsing of the input for your program then 'scanf' is the best approach to take.

You give scanf the format of the input you wish to parse and it will attempt to parse it converting integers, floats, characters, and strings for you and placing them into variables your code points scanf to. scanf returns the number of items successfully matched (or EOF at the end of the file).

The below code sums a list of integers.

```
#include<stdio.h>

int main() {
   int value;
   int sum = 0;
   while (scanf("%d", &value) == 1) {
      sum = sum + value;
   }
   printf("sum:%d\n",sum);
   return 0;
}
```

Exersizes

- Write a program that is given a list of x y coordinates via standard input and calculates the total distance traveled. This assumes you travelled in a straight line between coordinates.

If you ran the program with the file "t.txt" you would see the below:

```
ericm@lt:~/courses/ics/notes/code13simpleio$ cat t.txt
0.0 10.0
0.0 20.0
10.0 20.0
40.0 60.0

ericm@lt:~/courses/ics/notes/code13simpleio$ ./travel < t.txt
total distance:70.000000
```