



Command Line Parameters



Passing Unix parameters to C (1)

- ◆ Often a user wants to pass parameters into the program from the UNIX prompt

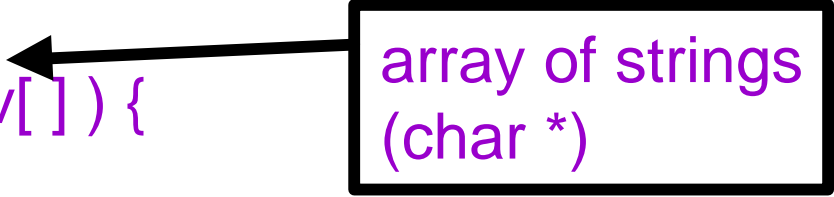
```
obelix[3] > progname arg1 arg2 arg3
```

- ◆ This is accomplished in C using `argc` and `argv`

- For example:

```
int main (int argc, char *argv[] ) {  
    /* Statements go here */  
}
```

array of strings
(char *)



Number of arguments



- ◆ Call this program from Unix

```
obelix [4] > progname arg1 "second arg" arg3
```

Passing Unix parameters to C (2)

```
#include <stdio.h>
int main(int argc, char *argv[]) {
    int count;
    printf ("Program name: %s\n", argv [0]);
    if (argc > 1) {
        for (count=1; count<argc; count++)
            printf ("Argument %d: %s\n",count,argv[count]);
    }
    else
        puts ("No command line arguments entered.");
    return 0;
}
```

Passing Unix parameters to C (3)

- ◆ Suppose we compiled the previous program to the executable file `myargs`
- ◆ `obelix[5] > myargs first "second arg" 3 4 > myargs.out`
- ◆ `myargs.out` contains the following lines:

Program name: `myargs`

Argument 1: `first`

Argument 2: `second arg`

Argument 3: `3`

Argument 4: `4`

Passing Unix arguments to C (4)

```
/* show file */
#include <stdio.h>
int main(int argc, char *argv[ ])
{
    FILE *fp;  int k;
    if(argc !=2) {
        printf("Usage: %s filename\n",
              argv[0]);
        return 0;
    }
    if((fp=fopen(argv[1], "r"))==NULL){
        printf("Cannot open file!\n");
        return 1;
    }
}
```

```
while((k=fgetc(fp))!=EOF )
    fputc(k, stdout);
fclose(fp);
return 0;
}
```

- ◆ Generally a main function of a C program for Unix checks whether the arguments are valid and prints simple help information.