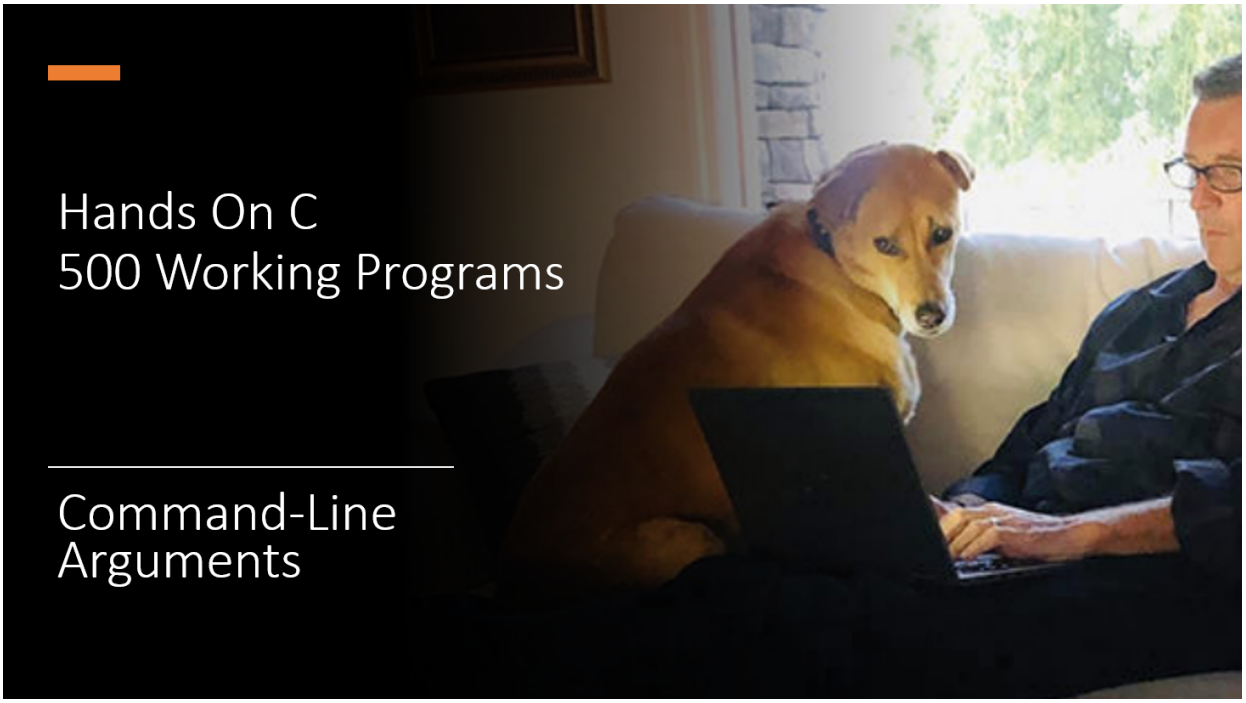

Understanding the Command Line

```
C:\> type Filename.txt  <Enter>

C:\> cd \home <Enter>

C:\> copy source.txt target.txt  <Enter>

C:\> del filename.ext  <Enter>
```

In [2]:
```c
#include <stdio.h>

int main(void)
{
   printf("Hello, world\n";
}
```

```
Hello, World!
```

In [2]:
```c
#include <stdio.h>

int main(int argc, char *argv[])
{
   printf("argc is %d\n", argc);
}
```

```
argc is 1
```

In [5]:
```c
#include <stdio.h>

int main(int argc, char *argv[])
{
   printf("%s\n", argv[0]);
}
```

```
/tmp/tmp_7m793ao.out
```

In [6]:
```c
#include <stdio.h>

int main(int argc, char *argv[])
{
   for (int i = 0; i < argc; ++i)
     printf("argv[%d] %s\n", i, argv[i]);
}
```

```
argv[0] /tmp/tmp1h3827j2.out
```

Using **argv to Access Command-Line Arguments

In [7]:
```
#include <stdio.h>

int main(int argc, char **argv)
{
   for (int i  = 0; i < argc; ++i)
     printf("%s\n", *argv);
}
```

```
/tmp/tmpickexxfm.out
```

In [8]:
```
#include <stdio.h>

int main(int argc, char **argv)
{
   while (*argv)
     printf("%s\n", *argv++);
}
```

```
/tmp/tmpfipnsdkm.out
```

There's Nothing Special About the Names argc and argv

In [10]:
```
#include <stdio.h>

int main(int a, char *b[])
{
   for (int i = 0; i < a; ++i)
     printf("a[%d] %s\n", i, b[i]);
}
```

```
a[0] /tmp/tmpryrxla46.out
```

Displaying the File Specified in the Command Line

In [12]:
```
#include <stdio.h>

int main(int argc, char *argv[])
{
    if (argc < 2)
      printf("Syntax: No file specified\n");
    else
    {
        FILE *fp = fopen(argv[1], "r");

        if (fp == NULL)
          printf("Error opening %s\n", argv[1]);
        else
        {
            char buffer[256];

            while (fgets(buffer, sizeof(buffer), fp))
                fputs(buffer, stdout);

            fclose(fp);
        }
    }
}
```

```
Syntax: No file specified
```

Deleting the File Specified in the Command Line

In [13]:
```
#include <stdio.h>

int main(int argc, char *argv[])
{
    if (argc < 2)
      printf("Syntax: No file specified\n");
    else if (remove(argv[1]) != 0)
      printf("Error deleting %s\n", argv[1]);
    else
      printf("%s deleted\n", argv[1]);
}
```

```
Syntax: No file specified
```

Displaying the First n Lines of the File Specified in the Command Line

```
Syntax
   first filename.ext 5  <Enter>

   first filename.ext  <Enter>
```

In [24]:
```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
 {
   char line[255];
   int i, j;

   FILE *fp;

   if (argc < 3)
     printf("Syntax: first filename [n]");
   else if ((fp = fopen(argv[1], "r")) == NULL)
     printf("Error opening %s\n", argv[1]);
   else
   {
      j = atoi(argv[2]);

      for (i=0; i < j; i++)
       {
         if (fgets(line, sizeof(line), stdin) == NULL)
            break;
         fputs(line, stdout);
       }
     fclose(fp);
   }
 }
```

```
Syntax: first filename [n]
```

Renaming the File Specified in the Command Line

```
Syntax
   Rename oldname.ext newname.ext
```

In [26]:
```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
 {
   if (argc < 3)
     printf("Syntax: rename filename newname\n");
   else if (rename(argv[1], argv[2]) != 0)
     printf("Error renaming %s\n", argv[1]);
   else
     printf("File renamed\n");
}
```

```
Syntax: rename filename newname
```

Understanding main's Return Value

In [29]:
```c
#include <stdio.h>

int main(void)
{
  return(0);
}
```

In [30]:
```c
#include <stdio.h>

int main(void)
{
  return(255);
}
```

```
[C kernel] Executable exited with code 255
```

In [31]:
```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
 {
   if (argc < 3)
     {
       printf("Syntax: rename filename newname\n");
       return(1);
     }
   else if (rename(argv[1], argv[2]) != 0)
     {
       printf("Error renaming %s\n", argv[1]);
       return(2);
     }
   else
     {
       printf("File renamed\n");
       return(0);
     }
}
```

```
Syntax: rename filename newname

[C kernel] Executable exited with code 1
```