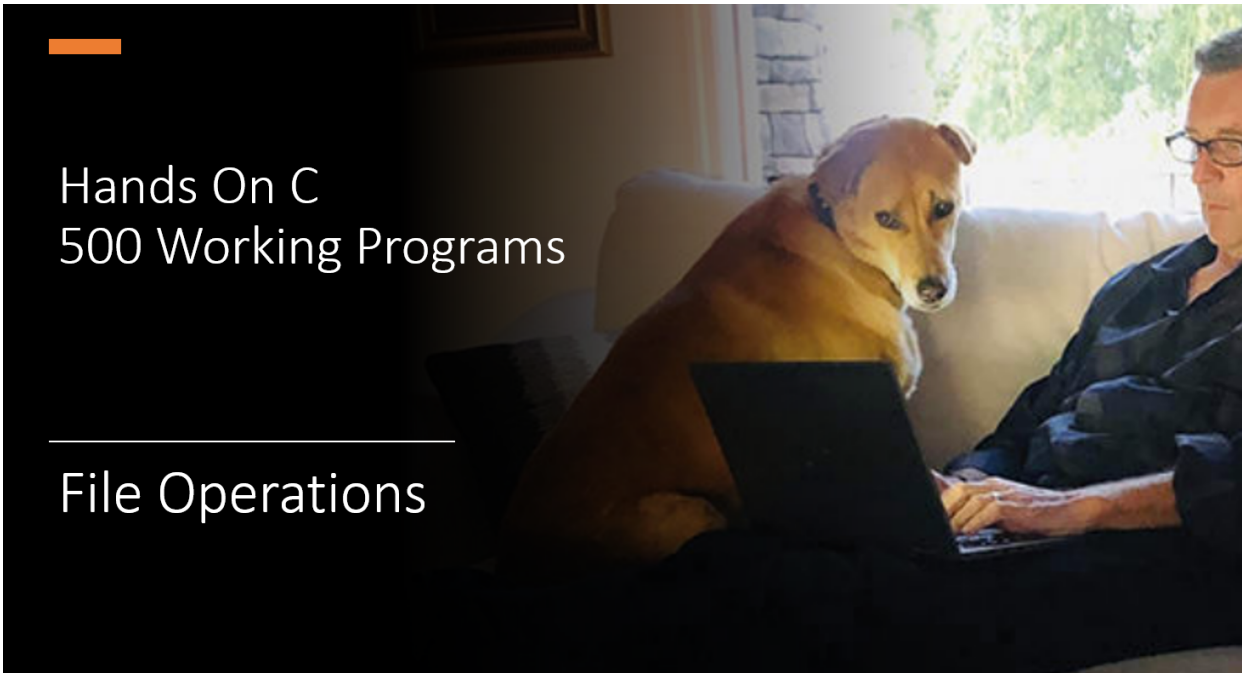

Understanding Files

```
Performing a File Operation

1. Declare a file pointer

       FILE *fp = fopen(filename, mode);

   Mode is "r" for read access, "w" for write access, "r+" for read/write
access, "w+" reading/writing, a+ append mode creating
   a new file if one did not exist
```

```
2. Perform the file operations

       fgets(buffer, sizeof(buffer), fp);

       fputs(buffer, fp);

3. Close the file

       fclose(fp);
```

In [1]:
```c
#include <stdio.h>

int main(void)
{
    FILE *fp = fopen("Output.txt", "w");

    fputs("Hello, world", fp);

    fclose(fp);
}
```

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

int main(void)
{
    FILE *fp = fopen("Output.txt", "r");
    char buffer[256];

    fgets(buffer, sizeof(buffer), fp);
    printf("Data read from the file: %s", buffer);
}
```

```
Data read from the file: Hello, world
```

Writing Output and Reading Input to/from a File a Character at a Time

In [3]:
```c
#include <stdio.h>

int main(void)
{
   FILE *fp = fopen("Alphabet.txt", "w");

   for (char letter = 'A'; letter <= 'Z'; letter++)
     fputc(letter, fp);

   fclose(fp);
}
```

In [4]:
```c
#include <stdio.h>

int main(void)
{
   FILE *fp = fopen("Alphabet.txt", "r");
   char letter;

   while ((letter = fgetc(fp)) != EOF)
     putchar(letter);

   fclose(fp);
}
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Writing/Reading Formatting Output to/from a File

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

int main(void)
{
    FILE *fp = fopen("Sample.txt", "w");

    fprintf(fp, "%s %d %f", "Kris", 50, 100000.0);

    fclose(fp);
}
```

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

int main(void)
{
    FILE *fp = fopen("Sample.txt", "r");
    char name[256];
    int age;
    float salary;


    fscanf(fp, "%s %d %f", name, &age, &salary);

    printf("Data read from file: %s %d %f\n", name, age, salary);
}
```

```
Data read from file: Kris 50 100000.000000
```

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

int main(void)
{
    FILE *fp = fopen("Sample.txt", "w");

    fprintf(fp, "%s %d %f", "Kris Jamsa", 50, 100000.0);  // White space error

    fclose(fp);
}
```

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

int main(void)
{
    FILE *fp = fopen("Sample.txt", "r");
    char name[256];
    int age;
    float salary;

    fscanf(fp, "%s %d %f", name, &age, &salary);         // white space error

    printf("Data read from file: %s %d %f\n", name, age, salary);
}
```

```
Data read from file: Kris 421308216 0.000000
```

Checking if a File Exists

In [9]:
```
#include <stdio.h>

int main(void)
{
    FILE *fp = fopen("Sample.txt", "r");

    if (fp != NULL)
      printf("File exists");

    fclose(fp);
}
```

```
File exists
```

Deleting a File

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

int main(void)
{
    int status = remove("Sample.txt");

    if (! status)
      printf("File Deleted");
}
```

```
File Deleted
```

Renaming a File

In [11]:
```c
#include <stdio.h>

int main(void)
{
    int status = rename("Output.txt", "Output.bak");

    if (! status)
      printf("File renamed\n");
    else
      printf("Error renaming file\n");
}
```

```
File renamed
```

Writing Structured Data to a File

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

struct Employee {
  int EmployeeID;
  char EmployeeName[256];
  char JobTitle[256];
  float Salary;
  char Phone[32];
};

void StoreStaff(struct Employee *worker)
{
    FILE *fp = fopen("Employee.dat", "wb");

    fwrite(worker, sizeof(struct Employee), 1, fp);

    fclose(fp);
}

int main(void)
{
  struct Employee coder = { 1, "Kris Jamsa", "Programmer", 100000, "928-555-1212"

  StoreStaff(&coder);
}
```

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

struct Employee {
  int EmployeeID;
  char EmployeeName[256];
  char JobTitle[256];
  float Salary;
  char Phone[32];
};

void LoadStaff(struct Employee *worker)
{
    FILE *fp = fopen("Employee.dat", "rb");

    fread(worker, sizeof(struct Employee), 1, fp);

    fclose(fp);
}


void ShowEmployee(struct Employee worker)
{
    printf("ID %d Name: %s Title %s Salary %6.2f Phone %s\n",
          worker.EmployeeID, worker.EmployeeName, worker.JobTitle, worker.Salary,
}


int main(void)
{
  struct Employee coder;

  LoadStaff(&coder);
  ShowEmployee(coder);
}
```

```
ID 1 Name: Kris Jamsa Title Programmer Salary 100000.00 Phone 928-555-1212
```

Testing a File Stream for Errors

In [14]:
```c
#include <stdio.h>

int main(void)
{
    FILE *fp = fopen("Alphabet.txt", "r");

    if (fp == NULL)
    {
      printf("Error opening file\n");
      return(-1);
    }
    else
    {
        char letter;

        while ((letter = fgetc(fp)) != EOF)
          if (ferror(fp))
          {
              printf("Error reading file\n");
              return(-1);
          }
        else
            putchar(letter);

        fclose(fp);
    }
}
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Creating and Using Temporary File

In [15]:
```c
#include <stdio.h>

int main(void)
{
    FILE *tp = tmpfile();

    fputs("ABCDE", tp);
    fputs("FGHIJ", tp);

    fseek(tp, 0, SEEK_SET);  // move to start of file

    char letter;

    while ((letter = fgetc(tp)) != EOF)
      putchar(letter);

    fclose(tp);
}
```

What You will Learn Next

```
When your run a program from a system prompt, the command you type is called
the command line.

C:\> COPY source.ext source.bak

To allow your programs to access the command line arguments, you can use two
parameters the operating system passes to main:

int main(int argc, char *argv[])

In the previous command line:

argc = 3
argv[0] = copy.exe
argv[1] = source.ext
argv[2] = source.bak
```