الجزء رقم (12) من سلسلة البرمجة المفتوحة بلغة C باستخدام Turbo CPP 3.0

هذه النسخة بتاريخ: 2007/11/12

برمجة الملفات

باستخدام TURBO C PLUS PLUS 3.0

برمجة: البراء عبد الرؤوف الرملي

طرابلس / ليبيا

Software Bara Ramli (SBR)

هذا الكتاب مجاني

مقدمة

في هذا الكتاب شرحت برمجة الملفات , والعلم فقد استفدت من كتاب:

Teach Yourself C in 21 Days

فقد وجدته نافعا مختصرا قد جمع معلومات قيمة , فاعتمدت منهجه في عرض المعلومات , أرجو الله أن ينفع به وأن يكون مساهمة منا في إثراء المكتبة العربية والمبرمج العربي.

ملاحظة: المكتبات والبرامج المعروضة , مرفقة مع الكتاب في مجلد (المرفقات).

وأريد أن أنبه على أن هذه المكتبة لازالت تحتاج إلى تطوير وإضافات حتى تكون مفيدة, وهذا يقع على عاتقنا جميعا حتى نصل بها إلى المستوى المطلوب , لذا فهي الآن بين يديك لتضيف إليها ما تظن أنه يرقى بها إلى المطلوب ثم تقوم بنشرها لتعم الفائدة لنا جميعا , لأنه ما لم نتشارك بأفكارنا , فلن نتقدم خطوة إلى الأمام.

البراء عبد الرؤوف الرملي
opencpp@yahoo.com
طرابس/ليبيا

يمكنك زيارة موقعي: www.khayma.com/opencpp

أغلب البرامج التي تكتب تستعمل الملفات لعدة أسباب منها :
تخزين البيانات , ترتيب المعلومات , وغيرها.

سيكون محور حديثنا عن:

- أنواع الملفات
- فتح ملف
- قراءة بيانات من ملف
- كتابة بيانات في ملف
- إغلاق ملف
- إدارة الملفات
- استعمال الملفات المؤقتة

أنواع الملفات:

بما أن الملفات streams تأتي على صورتين هما text و binary , لذا فمن المهم معرفة الفرق بينهما , حتى نستعمل النمط المناسب.

أولا text-streams:

إن text-streams هي ملفات ذات نمط نصي text-mode , والملفات ذات النمط النصي تحتوي على سلسلة من الأسطر.
كل سطر يبدأ بحرف أو أكثر "وقد لا يحتوي على أي حرف" , كما ينتهي برمز يشير إلى نهاية السطر end-of-file.
عدد الحروف والرموز المتوفرة هو 255 , من المهم التذكر بأن السطر line ليس string , لأنه لا يوجد له نهاية مفرغة للسطر أي NULL , بعبارة أخرى لا يوجد رمز '\0' في نهاية السطر.

إذا كيف يقوم نظام التشغيل operating system بإنشاء نهاية للسطر end-of-line داخل الملف؟

عندما تستخدم الملفات ذات النمط النصي فإن الذي يقوم به النظام هو ترجمة للرمز '\n' (الذي يعني بلغة C نهاية سطر) إلى رمز آخر (يختاره النظام للدلالة على نفس المعنى) , فمثلا: في نظام التشغيل DOS يكون رمز نهاية السطر هو (CR-LF) , اختصار لـ (carriage-return line-feed).

بعبارة أخرى عندما يقوم النظام بكتابة البيانات داخل ملف ذات نمط نصي فإن كل '\n' تترجم إلى CR-LF , بينما عند القراءة من ملف فإن كل CR-LF تترجم إلى '\n' , أما في نظام Unix فلا تحدث ترجمة بل تبقى '\n' دون تغيير.

ثانيا binary-streams:

أما binary-streams فترتبط بملفات ذات نمط ثنائي binary-mode , وجميع البيانات تكتب وتقرأ , وبدون ترجمة لرمز نهاية السطر أو رمز بداية سطر جديد.
وبذلك فـ new-line و NULL ليس لها وضعية خاصة وإنما تعامل كأي byte من البيانات.

ملاحظة: بعض دوال الإدخال والإخراج تستخدم للتعامل مع ملفات من نمط معين فقط , بينما تستخدم دوال أخرى أكثر من نمط.

أسماء الملفات:

يحتوي كل ملف في الجهاز على اسم , ويتحتم عليك استخدام أسماء للملفات كي تتمكن من التعامل معها , وللعلم فإن أسماء الملفات تخزن على هيئة string مثلها في ذلك مثل أي نص.

ما هي الصيغة العامة لـ "أسماء الملفات"
وهل تختلف باختلاف نظام التشغيل؟

في نظامي DOS و Windows 3.x كان اسم الملف يقبل مابين 1 إلى 8 حروف , بينما امتداد الملف يحتوي على 3 حروف , بخلاف Window 95 و Windows NT فقد كانا كأغلب أنظمة Unix يقبل 256 حرف.
أنظمة التشغيل تختلف حتى في أنواع الحروف المسموح باستخدامها في تسمية الملفات , ففي Windows 95 كانت الحروف (أو الرموز) الممنوعة هي:

| < > " ? * : \ /

لذلك يجب أن تكون مدرك لخصائص النظام الذي تستخدمه.
كما أن اسم الملف في لغة الـ C يشتمل أيضا على مسار الملف (الذي يشير إلى موقع الملف داخل القرص الصلب) , ويحدد في المسار القرص المستخدم وسلسلة المجلدات التي يقع داخلها الملف.

ما الذي يحدث إذا تم كتابة اسم الملف دون تحديد مساره؟

يقوم النظام باعتبار أن موقع الملف هو الموقع الافتراضي الحالي للنظام.

مثال على المسارات في نظامي DOS و Windows :

C:\data\list.txt

ملاحظة: تستخدم الشرطة المائلة للفصل بين المجلدات وغيرها , فمثلا: يقع الملف list.txt داخل المجلد data الموجود في القرص الصلب C , ويجب أن تتذكر أن:

الشرطة المائلة في لغة C إذا تم استخدامها لوحدها في string فلها معنى خاص , لذا يجب أن تسبقها واحدة أخرى , ويتم تمثيل المسار في لغة الـ C كالتالي:

```
char *filename = "c:\\data\\list.txt";
```

أما إذا قمت بإدخال المسار أثناء تنفيذ البرنامج بأحد دوال الإدخال فتكتفي بشرطة مائلة واحدة فقط.

عند كتابة المسار ليس جميع الأنظمة تستخدم (\) الشرطة المائلة "ناحية اليسار" , فمثلا نظام Unix يستخدم (/) الشرطة المائلة "ناحية اليمين".

فتح ملف:

عند فتح الملف يصبح جاهزا للقراءة (أي انتقال البيانات من الملف إلى البرنامج) و الكتابة (حفظ بيانات البرنامج داخل الملف), أو كليهما , ويلزمك إغلاق الملف عند انتهائك من استعماله.

نستخدم الدالة fopen لفتح ملف , وهي معرفة داخل المكتبة stdio.h وتكون صيغتها العامة كالتالي:

```
FILE *fopen(filename, mode);
```

أي أن الدالة ترجع قيمة مؤشر (pointer) من نوع FILE , لذا يلزمك عند فتح ملف للقراءة أن تعرف مؤشر (pointer) من نوع FILE , وتقوم الدالة بإرجاع قيمة NULL إذا فشلت في تنفيذ المطلوب , فمثلا عند فتح ملف من قرص مرن (flopy) لم تتم تهيئته (format) سابقا.

المتغير filename هو مصفوفة نصية تدل على مسار الملف , بينما المتغير mode تدل على صيغة فتح الملف (قراءة أو كتابة أو كليهما) , ويتم تخصيص رمز معين للمصفوفة , كل رمز يدل على حالة , كالتالي:

mode	فتح ملف
r	للقراءة لن تخسر البيانات السابقة ملاحظة: إذا كان الملف غير موجود فسترجع الدالة fopen قيمة NULL
w	للكتابة ستخسر البيانات السابقة ملاحظة: إذا كان الملف غير موجود فسيتم إنشاء ملف جديد.
a	للكتابة لن تخسر البيانات السابقة (ستتم إضافة البيانات في نهاية ملف) ملاحظة: إذا كان الملف غير موجود فسيتم إنشاء ملف جديد.
r+	للقراءة والكتابة عند القراءة لن تخسر البيانات السابقة عند الكتابة ستخسر البيانات السابقة ملاحظة: إذا كان الملف غير موجود فسترجع الدالة fopen قيمة NULL
w+	للقراءة والكتابة عند القراءة أو الكتابة: ستخسر البيانات السابقة ملاحظة: إذا كان الملف غير موجود فسيتم إنشاء ملف جديد.
a+	للقراءة والكتابة عند القراءة: لن تخسر البيانات السابقة عند الكتابة: لن تخسر البيانات السابقة (ستتم إضافة البيانات في نهاية ملف) ملاحظة: إذا كان الملف غير موجود فسيتم إنشاء ملف جديد.

كل السابق للتعامل مع الملفات ذات النمط النصي أما للتعامل مع النمط الثنائي فقم بإضافة حرف a أو حرف b للقيم السابقة.

تذكر أن الدالة fopen ترجع قيمة NULL عند حدوث خطأ , يتضمن التالي:

- استخدام اسم ملف غير صالح
- محاولة فتح ملف من قرص غير جاهز أو غير متوفر

- محاولة فتح ملف من مسار (directory) غير موجود
- محاولة فتح ملف غير موجود , (عند استخدام صيغة r في الـ mode).

على أية حال , تحتاج قبل استعمال الملف التأكد من عدم حدوث خطأ أثناء فتحه , في الحقيقة لن نتأكد من سبب الخطأ بالضبط ولكن تستطيع إظهار رسالة تنبيه للمستخدم كي يحاول فتح الملف من جديد , أو يمكنك إنهاء البرنامج.

أغلب (compiler) المستخدم للغة C يتضمنون (NON-ANSI) لعرض جوهر الخطأ الحاصل.

تعريف المكتبات	#include <stdlib.h> #include <stdio.h>
تعريف مؤشر fp للملف تعريف مصفوفة filename للدلالة على اسم الملف , ومصفوفة mode للدلالة على صيغة الفتح	void main() { FILE *fp; char filename[40], mode[4];
لإدخال اسم الملف	printf("\nEnter a filename: "); gets(filename);
لإدخال صيغة الفتح	printf("\nEnter a mode, max 3 characters:"); gets(mode);

إذا تم الفتح بنجاح
فسيطبع رسالة تفيد بذلك
ومن ثم يغلق الملف
	if ((fp = fopen(filename, mode)) != NULL)
	{
	printf("\nSuccessful opening %s ", filename);
	printf("in mode %s.\n", mode);
	fclose(fp);
	}

إذا حدث خطأ
فسيطبع رسالة تفيد بذلك
	else
	{
	printf("\nError opening file %s ", filename);
	printf("in mode %s.\n", mode);
	}

	}

الكتابة والقراءة داخل ملف:

البرنامج الذي يستعمل الملفات بإمكانه الكتابة داخل ملف والقراءة من ملف , كما بإمكانه المقارنة بين ملفين.

بإمكانك الكتابة داخل ملف بثلاثة طرق:

- **ملفات عامة لحفظ قيم عددية ونصوص , وبإمكانك فتحها ببرنامج آخر**
- **ملفات حرفية لحفظ عدة حروف أو عدة أسطر , مثل معالج الكلمات**
- **ملفات مباشرة لتخزين أجزاء من الذاكرة مباشرة إلى الملف**

ملاحظة: يجب عليك قراء الملف بنفس نوع النمط الذي كتب به , ولكن هذه ليست قواعد فلغة الـ C لغة مرنة (وهذه أحد مميزاتها) , فالمبرمج الماهر يستطيع استعمال أي نوع ويعالجه حسب احتياجاته , ولكن المبتدئ يتبع أسهل الطرق حسب منهج ميسر.

الملفات العامة:

الملف العامة تقبل القيم النصية والرقمية بطرق خاصة , تشبه إدخال لوحة المفاتيح وإخراج الشاشة مثل printf() و scanf()

دوال الكتابة في ملف:

دالة الإخراج هي fprintf وهي معرفة في المكتبة stdio.h مثلها في ذلك مثل دالة الإدخال printf غير أن الدالة fprintf نكتب فيها مؤشر الملف.

شكل الدالة كالتالي:

fprintf(pf, form, a, b, c, ...);

حيث pf مؤشر للملف, وهو يحمل قيمة الدالة fopen , أما الباقي فمثل دالة printf فـ form هي الصيغة المراد طباعتها بينما a,b,c... هي المتغيرات المراد طباعتها داخل ملف.

تعريف المكتبات	`#include <stdlib.h>` `#include <stdio.h>`
	`void clear_kb(void);`
	`main()` `{`

تعريف مؤشر fp للملف
تعريف مصفوفة filename للدلالة على اسم الملف , ومصفوفة mode للدلالة على صيغة الفتح
	`FILE *fp;`
	`float data[5];`
	`int count;`
	`char filename[20];`

لإدخال 5 قيم	puts("Enter 5 floating-point numerical values.");
	for (count = 0; count < 5; count++)
	scanf("%f", &data[count]);

كي نتمكن من استخدام دالة gets	clear_kb();
لإدخال اسم الملف (المسار)	puts("Enter a name for the file."); gets(filename);

إذا حدث خطأ أثناء فتح الملف
فسيطبع رسالة تفيد بذلك
ومن ثم يخرج من البرنامج
	if ((fp = fopen(filename, "w")) == NULL)
	{
	printf("Error opening file %s.", filename);
	exit(1);
	}

طباعة القيم المدخلة سابقا في الملف fp
وطباعة القيم أيضا على الشاشة
	for (count = 0; count < 5; count++)
	{
	fprintf(fp, "\ndata[%d] = %f", count, data[count]);
	printf("\ndata[%d] = %f", count, data[count]);
	}

إغلاق الملف	fclose(fp);
	return(0);
	}

	void clear_kb(void)
	{
	char junk[80];
	gets(junk);
	}

دوال القراءة من ملف:

دالة الإخراج هي fscanf وهي معرفة في المكتبة stdio.h مثلها في ذلك مثل دالة الإدخال scanf غير أن الدالة fscanf نكتب فيها مؤشر الملف.

شكل الدالة كالتالي:

fscanf(pf, form, a, b, c, ...);

حيث pf مؤشر للملف, وهو يحمل قيمة الدالة fopen , أما الباقي فمثل دالة scanf فـ form هي الصيغة المراد طباعتها بينما a,b,c... هي المتغيرات المراد قراءتها من الملف

ملاحظة: تحتاج لملف به قيم لقراءتها , فقم بكتابة خمسة قيم واجعل بينهم مسافة فارغة أو حتى سطر جديد فيكون شكل القيم كالتالي:

```
2.55  1.02
22.47
48.09 7.98
```

ومن ثم نشغل البرنامج التالي:

تعريف المكتبات	#include <stdlib.h> #include <stdio.h>
تعريف 5 متغييرات من نوع عشري تعريف مؤشر fp للملف	void main() { float f1, f2, f3, f4, f5; FILE *fp;

اختبار ما إذا كان الملف موجودا
إذا حدث خطأ أثناء فتح الملف
فسيطبع رسالة تفيد بذلك
ومن ثم يخرج من البرنامج
	if ((fp = fopen("C:\\W.TXT", "r")) == NULL)
	{
	printf("Error opening file.\n");
	exit(1);
	}

قراءة 5 قيم من الملف وتخزينهم في المتغييرات من نوع float	fscanf(fp, "%f %f %f %f %f", &f1, &f2, &f3, &f4, &f5);
طباعة الـ 5 متغييرات	printf("The values are %f, %f, %f, %f, and %f", f1, f2, f3, f4, f5);
إغلاق الملف	fclose(fp);
	}

ملاحظة: القيم التي ستطبع قد لا تكون بنفس الدقة الموجودة داخل الملف
فمثلا رقم 100.02 قد يظهر 100.01999

الملفات الحرفية:

يمكن إدخال سلسلة من الحروف تمثل نص.

دوال القراءة من ملف:

هناك 3 دوال لإدخال الحروف في ملف هي: getc وfgetc بينما تستخدم fgets لإدخال سطر من الأحرف.

الدالتين getc و fgetc توأمان وصيغة الأولى:

int getc(FILE *fp);

حيث pf مؤشر للملف, وهو يحمل قيمة الدالة fopen , تقوم الدالة بكتابة حرف داخل ملف , وإن حدث خطأ فترجع EOF

لابد أنك لاحظت أن نفس دوال الإدخال من لوحة المفاتيح تستخدم القراءة من ملف وهذه ميزة من مميزات لغة C

إذا كانت الدالتين getc و fgetc ترجعان حرف واحد فلماذا كتب في تعريف الدالة أنها ترجع عدد صحيح؟

سبب ذلك يرجع لأنه عند قراءة الملف لابد من معرفة مؤشر نهاية الملف وهذا المؤشر قيمته ليست حرفية بل عدد صحيح.

لإدخال سطر من الحروف نستخدم الدالة fgets

```
char *fgets(char *str, int n, FILE *fp);
```

حيث pf مؤشر للملف, وهو يحمل قيمة الدالة fopen() , وحيث n هو الحد الأقصى لعدد الحروف المسموح به , بينما str مصفوفة تخزن فيها الحروف التي تمت قراءتها.

عند استدعاء دالة fgets تقوم بقراءة حرف من الملف fp إلى الذاكرة في المصفوفة str أما العدد الصحيح n فيسمح بعدد بقراءة عدد معين من الحروف زمن ثم يضيف رمز '\0' للدلالة على نهاية النص , وإذا نجحت الدالة فإنها ترجع النص إلى str , وفي حالة حدوث خطأ فإن الدالة ترجع قيمة NULL وقد يكون سبب ذلك:

\\\\\\\\\\\\\\\\\\\\\\\\

\\\\\\\\\\\\\\\\\\\\\\\\

دوال الكتابة في ملف:

هناك دالتين لكتابة الحروف في ملف هي: putc و fputs

أولا الدالة: putc ووظيفتها كتابة حرف في ملف معين , وهي معرفة داخل المكتبة stdio.h وصيغتها العامة:

```
int putc(int ch, FILE *fp);
```

حيث pf مؤشر للملف, وهو يحمل قيمة الدالة fopen , تقوم الدالة بكتابة حرف في ملف , وإن حدث خطأ فترجع EOF وهو يعرف بقيمة -1 في مكتبة stdio.h

ثانيا الدالة: fputs ووظيفتها كتابة سطر من الأحرف موجودة في ملف معين , وهي معرفة داخل المكتبة stdio.h وصيغتها العامة:

```
int fputs(char *str, FILE *fp);
```

وهي تعمل مث الدالة puts إلا أن الفرق الوحيد بينهما أن دالة fputs يمكنك فيها أن تحدد الملف المراد كتابة البيانات فيه.

وهي لا تضيف رمز سطر جديد في نهاية النص فإن أردت ذلك فقم بكتابته.

حيث pf مؤشر للملف, وهو يحمل قيمة الدالة fopen , تقوم الدالة بكتابة سطر من الأحرف str الذي ينتهي بـ '\0' وترجع الدالة قيمة موجبة إذا نجحت , وإن حدث خطأ فترجع EOF وهو يعرف بقيمة 1- في مكتبة stdio.h

الملفات المباشرة:

وهو يستخدم لقراءة رسالة من برنامج بلغة C أو غيرها , ويستخدم في النمط الثنائي فقط , وعند كتابة البيانات داخل ملف فإن البيانات تكتب من الذاكرة إلى الملف , والقراءة عملية عكسية.

دوال الكتابة في ملف:

دالة الإخراج هي fwrite لتخزين بيانات من الذاكرة إلى ملف ثنائي , وهي معرفة في المكتبة stdio.h على الشكل التالي:

```
int fwrite(void *buf, int size, int count, FILE *fp);
```

buf هو المعلومات المراد طباعتها

فمثلا لتخزين مصفوفة من 100 عنصر من نوع int فنجعل size=2 لأن كل عدد int=2 بايت

Count هو عدد العناصر المراد طباعتها وهي هنا 100 عنصر

حيث pf مؤشر للملف, وهو يحمل قيمة الدالة fopen() في حالة نجاح الدالة فهي ترجع عدد العناصر التي تم طباعتها وبذلك يمكنك التأكد هل جميع العناصر تم طباعتها أو لا باستخدام الأمرين:

```
if( (fwrite(buf, size, count, fp)) != count)
printf("Error writing to file.");
```

مثال: لطباعة متغير x من نوع double

```
double x=10.00;
fwrite(&x, sizeof(double), 1, fp);

fwrite(data, sizeof(address), 50, fp);
fwrite(data, sizeof(data), 1, fp);
```

دوال القراءة من ملف:

دالة الإدخال هي fread() لتخزين بيانات من ملف ثنائي إلى الذاكرة, وهي معرفة في المكتبة stdio.h على الشكل التالي:

int fread(void *buf, int size, int count, FILE *fp);

#include <stdlib.h> #include <stdio.h> #define SIZE 20	

**main()
{
int count, array1[SIZE], array2[SIZE];
FILE *fp;
 	
/* Initialize array1[]. */
 	
for (count = 0; count < SIZE; count++)
array1[count] = 2 * count;
 	
/* Open a binary mode file. */
 	
if ((fp = fopen("direct.txt", "wb")) == NULL)
{
fprintf(stderr, "Error opening file.");
exit(1);
}
/* Save array1[] to the file. */
 	
if (fwrite(array1, sizeof(int), SIZE, fp) != SIZE)
{
fprintf(stderr, "Error writing to file.");
exit(1);
}
 	
fclose(fp);**	

**/* Now open the same file for reading in binary mode*/
 	
if ((fp = fopen("direct.txt", "rb")) == NULL)
{
fprintf(stderr, "Error opening file.");
exit(1);
}
 	
/* Read the data into array2[]. */
 	
if (fread(array2, sizeof(int), SIZE, fp) != SIZE)
{
fprintf(stderr, "Error reading file.");
exit(1);
}
 	
fclose(fp);**	

/* Now display both arrays to show they're	

```
the same. */

  for (count = 0; count < SIZE; count++)
     printf("%d\t%d\n", array1[count],
array2[count]);
  return(0);
 }
```

إغلاق الملفات :

int fclose(FILE *fp);
returns 0 on success or -1 on error

int fcloseall(void);

You can flush a stream's buffers without closing it by using the fflush() or flushall() library functions. Use fflush() when you want a file's buffer to be written to disk while still using the file. Use flushall() to flush the buffers of all open streams. The prototypes of these two functions are as follows:

int fflush(FILE *fp);
int flushall(void);

The argument fp is the FILE pointer returned by fopen() when the file was opened. If a file was opened for writing, fflush() writes its buffer to disk. If the file was opened for reading, the buffer is cleared. The function fflush() returns 0 on success or EOF if an error occurred. The function flushall() returns the number of open streams.

DO open a file before trying to read or write to it.

DON'T assume that a file access is okay. Always check after doing a read, write, or open to ensure that the function worked.

DO use the sizeof() operator with the fwrite() and fread() functions.

DO close all files that you've opened.

DON'T use fcloseall() unless you have a reason to close all the streams

إدارة الملفات :

حذف ملفات:

int remove(const char *_filename_);

ترجع 0 إذا أتمت العمل بنجاح وإذا حدث خطأ رجعت -1

#include <stdio.h>	

main()
{
char filename[80];
printf("Enter the filename to delete: ");
gets(filename);	

if (remove(filename) == 0)
printf("The file %s has been deleted.\n", filename);
else
fprintf(stderr, "Error deleting the file %s.\n", filename);	

return(0); }	

إعادة تسمية ملف:

int rename(const char *_oldname_, const char *_newname_);

ترجع 0 إذا أتمت العمل بنجاح وإذا حدث خطأ رجعت -1

#include <stdio.h>	

main()
{
char oldname[80], newname[80];
printf("Enter current filename: ");
gets(oldname);
printf("Enter new name for file: ");
gets(newname);	

if (rename(oldname, newname) == 0)
printf("%s has been renamed %s.\n", oldname, newname);
else
fprintf(stderr, "An error has occurred renaming %s.\n", oldname);	

return(0); }	

نسخ الملفات:

#include <stdio.h> int file_copy(char *oldname, char *newname);	

main()
{
char source[80], destination[80];
 	
/* Get the source and destination names. */
 	
printf("\nEnter source file ");	

```
    gets(source);
    printf("\nEnter destination file ");
    gets(destination);

    if ( file_copy( source, destination ) == 0 )
        puts("Copy operation successful");
    else
        fprintf(stderr, "Error during copy
operation");
    return(0);
 }
```

```
int file_copy( char *oldname, char *newname
)
 {
    FILE *fold, *fnew;
    int c;

    /* Open the source file for reading in
binary mode. */

    if ( ( fold = fopen( oldname, "rb" ) ) ==
NULL )
        return -1;

    /* Open the destination file for writing in
binary mode. */

    if ( ( fnew = fopen( newname, "wb" ) ) ==
NULL  )
    {
        fclose ( fold );
        return -1;
    }

    /* Read one byte at a time from the
source; if end of file */
    /* has not been reached, write the byte to
the */
    /* destination. */
```

```
    while (1)
    {
        c = fgetc( fold );

        if ( !feof( fold ) )
            fputc( c, fnew );
        else
            break;
    }

    fclose ( fnew );
    fclose ( fold );

    return 0;
 }
```

الملفات المؤقتة :

char *tmpnam(char *s);

DON'T remove a file that you might need again.

DON'T try to rename files across drives.

DON'T forget to remove temporary files that you create. They aren't deleted automatically.

الكتاب لم يكتمل بعد ولكن
هذا ماتيسر حتى الآن وسنكمل
ماتبقى في النسخ القادمة
إن شاء الله.