جامعة حماه

السنة الثالثة

قسم تقنيات الحاسوب

نظم تشغيل شبكية

المحاضرة الثانية

INPUT - OUTPUT

إعداد وتقديم

م.مها رضوان جحا م.مصطفى وهبة **م.فائزة حاتم**

إن Pipe هي حالة خاصة من إعادة التوجيه redirection حيث تقوم بتحويل خرج برنامج إلى دخل برنامج آخر .

و الآن سنأخذ إعادة الوجيه بمفهومها العام .

مثلا لو نفذنا التعليمة : $ls

سيتم طباعة الناتج على الخرج القياسي stdout و هو الشاشة (افتراضيا)

تحويل خرج البرنامج إلى ملف :

لتحويل خرج البرنامج إلى ملف نستخدم > .

```
$program  > file
```

مثال:

```
$ls  > myfile
```

مثال:

```
$ls  > myfile
```

تقوم هذه التعليمة بإنشاء الملف myfile و كتابة ناتج عملية ls إليه ,و إذا كان الملف myfile موجود مسبقا فتقوم بعملية over write للملف .

لو استعرضنا محتوى الملف: $cat myfile

سنرى ناتج عملية ls السابقة و لكن مع اختلاف بسيط في التنسيق ،حيث تظهر أسماء الملفات مثلا بدون ألوان ،ذلك لأننا لا نقوم بنسخ ما هو موجود على الشاشةإلى الملف و إنما نحول خرج ls إلى الملف myfile (بدلا من stdout).

- لإضافة خرج البرنامج إلى نهاية الملف : >>

```
$program  >>file
```

مثال:

نريد الاحتفاظ بتاريخ العملية التي نقوم بها في ملف عن طريق التعليمة uptime التي تعطي مجموعة من المعلومات مثلا : الزمن بدءا من لحظة تشغيل الحاسب،عدد المستخدمين ،المستخدم الحالي و الـ shell التي يستخدمها ...

للاحتفاظ بهذه المعلومات في ملف :

$uptime >timefile ⟶ over write
$uptime >>timefile ⟶ append

مثال:

البحث عن ملف Network بدءا من /etc

```
$find /etc -type f -name Network
```

تظهر هنا بعض رسائل الخطأ (permission denied)أي أن الوصول إلى بعض الملفات غير مسموح للمستخدم الحالي .

في الحقيقة إن خرج هذه التعليمة يظهر على ملفين (رغم أن الطباعة تتم على الشاشة في الحالتين) و هما :

Stdout ⟶ الخرج الطبيعي
Stderr ⟶ Permission denied

في الحالة الافتراضية stdout و stderr على الشاشة.

لتحويل خرج التعليمة السابقة إلى ملف :

```
$find /etc -type f -name Network > outfile
```

هنا قمنا بتحويل الخرج stdout فقط ،و ليس خرج الشاشة..!!

لذلك ستلاحظ أنه ما زالت تظهر رسائل الخطأ على الشاشة فقط (لأنها تظهر علـــى stderr و نحن قمنا بتحويل الخرج stdout فقط .

–تحويل الخرج stderr : 2>

يتم تحويل الـــ stderr عن طريق الواصف descriptor الذي تحدثنا عنه في المحاضرة الماضية :

$program 2> file

```
$find /etc -type f -name Network 2> errfile
```

هنا نلاحظ أنه لم تعد تظهر رسائل الخطأ على الشاشة و إنما يظهر فقط الخرج النظامي (stdout)

تحويل stderr إلى /dev/null :

إذا كانت لا تهمنا رسائل الخطأ (stderr) فإننا نقوم بتحويله إلى /dev/null و هو عبارة عن جهاز وهمي .

مثلا:

```
$find /etc -type f -name Network 2> /dev/null
```

يظهر على الشاشة stdout فقط .

بينما تم تحويل stderr إلى الملف /dev/null ،لو حاولنا استعراض محتوى هذا الملف :

```
$cat /dev/null
```

لا يظهر أي شيء على الشاشة لأنه أصلا ملف لحذف المعلومات التي لا نريدها (جهاز وهمي).

تحويل الدخل stdin : <

إن stdin الافتراضي هو الكيبورد .

مثال :

إن مهمة تعليمة cat الحقيقية هي كتابة stdin إلى stdout .

نفذ تعليمة cat دون تمرير معاملات لها .

```
$cat

Moustafa
Moustafa
```

عند تنفيذ cat دون تمرير اسم ملف لها ، فإنها تعتبر الدخل هو الدخل الافتراضي من الكيبورد ،حيث تطلب عملية قراءة من الملف stdin و تنتظر الدخل من الكيبورد لتقوم بطباعة الدخل على الشاشة (stdout افتراضيا).

تحويل الدخل من ملف :

```
$cat < myfile
```

الآن أصبحت cat تقرأ من stdin (و الذي أصبح الملف myfile بدلا من لوحة المفاتيح) و يقوم بالعرض على stdout .

ملاحظة :

التعليمة التي تقرأ من stdin يمكن تحويل دخلها ،لكن التعليمة التي لا تقرأ من stdin لا يمكن تحويل دخلها .

مثلا : ls لا تقرأ من stdin .

ملاحظة :

لا تخلط بين وظيفة البرنامج ،و تحويل الدخل أو الخرج لأن الوظيفة لا علاقة لها بالدخل و الخرج .

تحويل الدخل و الخرج :

```
$program  <infile   >outfile
```

مثال:

```
$cat  <file1   >file2
```

أصبح stdin هو file1 ، و stdout هو file2 .

– هذه العملية مشابهة لعملية copy من stdin إلى stdout .

diff :

```
$diff   file1   file2
```

و هو برنامج يظهر الاختلاف بين ملفين .

حيث يظهر الاختلاف من الملف الأول ثم الاختلاف من الملف الثاني .

– في حال لم يكن هناك أي أسطرة مشتركة بين الملفين فإنه يقوم باستعراض الملف الأول ثم استعراض الملف الثاني .

`command` : command substitution

عندما نضع أمر بالشكل : $`command`

فإنه ينفذ الأمر و يعامل ناتج التنفيذ كأمر (يحاول تنفيذه)

أي تقوم الـــ shell بحذف الفواصل و تنفيذ ما ضمنها ثم تعيد النتيجة إلى نفس المكان لتنفذها كأمر.

مثال :

عرض الملفات الموجودة في home و تنتهي بـــ txt.

```
$ls *.txt
```

و الآن نريد حذف هذه الملفات :

```
$ls *.txt | rm
```

لا يمكن لأن rm لا تأخذ دخل من stdin لذلك لن يتم الحذف .

نعالج هذه المشكلة كمايلي :

```
$rm  `ls  *.txt`
```

و بالتالي يتم تنفيذ تعليمة ls و إعادة الناتج إلى مكانها كدخل لـــ rm .