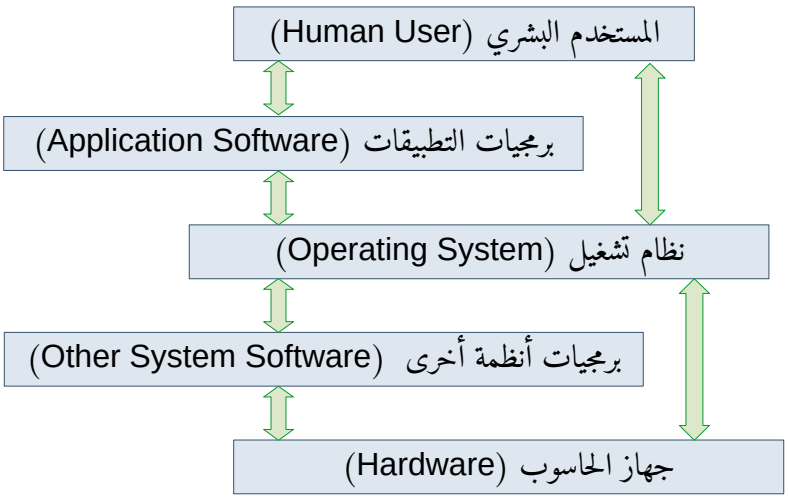
المحاضرة #٦: أنظمة التشغيل

١ مقدمة

تنقسم البرمجيات الحديثة إلى صنفين: برمجيات تطبيقات وبرمجيات أنظمة. تكتب **برمجيات التطبيقات** لتلبية احتياجات محددة ولحل المشكلات الحياتية. على سبيل المثال برامج معالجة النصوص والألعاب وأنظمة إدارة المخازن وبرامج تشخيص أعطال السيارات وبرامج قيادة الصواريخ. بينما تقوم **برمجيات النظام** بإدارة نظام الحاسوب على مستوى أولي. إذ توفر الأدوات والبيئة المناسبة لتكوين وتنفيذ برمجيات التطبيقات. غالبا ما تتفاعل برمجيات النظام بشكل مباشر مع أجهزة الحاسوب (Hardware) مما يوفر وظائف اكثر من أجهزة الحاسوب نفسها.

يمثل **نظام التشغيل** قلب برمجيات النظام في الحاسوب. يدير نظام التشغيل موارد الحاسوب مثل الذاكرة ووحدات الإدخال والأخراج ويوفر واجهة لتفاعل المستخدم مع الحاسوب. تدعم برمجيات الأنظمة الأخرى أغراض تطبيقات محددة مثل مكتبة البرمجيات الرسومية والمسؤولة عن تكوين الصور على الشاشة. يسمح نظام التشغيل للتطبيقات بالتفاعل مع موارد النظام الأخرى. الشكل التالي يوضح طريقة تفاعل نظام التشغيل مع بقية عناصر الحاسوب الأخرى.



شكل ١: تفاعل نظام التشغيل مع بقية عناصر الحاسوب

مصدر الشكل كتاب Computer Science Illuminated by Nell B. Dale

يدير نظام التشغيل موارد الحاسوب بحيث يسمح لبرمجيات التطبيقات الوصول إلى موارد الحاسوب أما مباشره أو عبر برمجيات أنظمة أخرى. كما يوفر واجهة للمستخدم للتفاعل مع نظام الحاسوب.

٢ أدارة الموارد (Resource Management)

عند تنفيذ البرنامَج سوف يستقر في الذاكرة الرئيسية لكي يبدأ المعالج بتنفيذ كل إيعازاته واحد تلو الأخر. أن **البرمجة المتعددة (Multiprogramming)** هي وجود عدة برامج في الذاكرة الرئيسية في نفس الوقت للمنافسة في الوصول إلى المعالج لغرض التنفيذ. جميع أنظمة التشغيل الحديثة تستخدم البرمجة المتعددة بدرجة أو بأخرى. لذلك يجب على نظام التشغيل **أدارة الذاكرة (Memory Management)** لغرض تتبع هذه البرامج وعنوان تواجدها فيها.

احد المفاهيم الرئيسية لأنظمة التشغيل هو فكرة **المعالجة (Process)** وهي البرنامَج في حالة التنفيذ. أن البرنامَج هو مجموعة من الإيعازات في حين أن المعالجة (Process) تمثل البرنامَج خلال عملية تنفيذه. قد تكون هنالك عدة معالجات (Processes) فعالة في نفس الوقت خلال البرمجة المتعددة وقد يتم مقاطعة (Interrupt) المعالجة (Process) الواحدة خلال تنفيذها. لذلك على نظام التشغيل **أدارة المعالجة (Process Management)** لتتبع تقدم المعالجة في كل حالاته.

أضافه إلى إدارة الذاكرة و إدارة المعالجة نحتاج إلى عملية **جدولة المعالج (CPU Scheduling)** وهي العملية المسؤولة عن تحديد إي من المعالجات المتواجدة في الذاكرة يجب تنفيذها من قبل المعالج في الوقت المناسب.

٣ أدارة الذاكرة (Memory Management)

كما ذكرنا أعلاه فان في بيئة البرمجة المتعددة تخزن عدة برامج مع بياناتها في الذاكرة في نفس الوقت لغرض تنفيذها من قبل المعالج. لذلك يجب على نظام التشغيل أن يستخدم تقنياته لإنجاز المهام التالية:

- تتبع مكان وكيفية تواجد البرنامَج في الذاكرة
- تحويل العناوين الافتراضية للذاكرة إلى عناوين حقيقية

عند كتابة إي البرنامَج بإحدى لغات المستوى الأعلى (High-level Language) مثل بايثون (Python) او سي++ (C++) نعين الكثير من القيم للمتغيرات (مثلاً i=1 او y=x+10) فعند تَرْجَمَة البرنامَج (إي تحويله إلى لغة الآله) يجب خزن هذه القيم في الذاكرة (حيث يخزن البرنامَج وبياناته فيها) ولكن تكمن المشكلة في إننا لا نعلم أين موقع الذاكرة الذي سوف تخزن فيه. كيف يمكننا تحديد العنوان لأي شيئ نرغب بخزنه في الذاكرة؟

أن حل هذه المشكلة يكمن في استخدام نوعين من العناوين: **العناوين الافتراضية (Logical Addresses)** و**العناوين الحقيقية (Physical Addresses)**. أن العنوان الافتراضي هو قيمة افتراضية لموقع البرنامَج وبياناته ولا عِلاقة له بالعنوان الحقيقي للذاكرة. أما العنوان الحقيقي فهو العنوان الحقيقي في رقاقة الذاكرة الرئيسية.

عندما يتم تَرْجَمَة البرنامَج فان المتغيرات وغيرها من القيم يتم تغيرها إلى عناوين افتراضية (Logical Addresses). وعندما يتم تحميل البرنامَج إلى الذاكرة (عند بدأ مرحلة تنفيذ البرنامج) فان كل عنوان افتراضي سوف يترجم إلى عنوان حقيقي (Physical Address).

٤ أنظمة المِلَفّات (File Systems)

في المحاضرة الثالثة حددنا الاختلافات بين الذاكرة الرئيسية والثانوية. حيث أن الذاكرة الرئيسية هي المكان الذي يتم فيه تخزين البرامج والبيانات النشطة أثناء تنفيذها. أن الذاكرة الرئيسية متقلبة (volatile) بمعنى أن البيانات المخزنة عليها سوف تُفقد إذا انقطع التيار الكهربائي. بينما الذاكرة الثانوية غير متقلبة (nonvolatile) إي يتم الاحتفاظ بالبيانات المخزنة عليها حتى في حالة انقطاع الطاقة. لذلك نستخدم الذاكرة الثانوية للتخزين الدائم لبياناتنا.

أن أكثر أجهزة التخزين الثانوية شيوعًا هو محرك الأقراص المغناطيسية (magnetic disk drive) وقرص الحالة الصلبة (solid state device). يشمل كلاً من الأقراص الخزن الصلبة الموجودة داخل الحاسوب والأقراص المحمولة التي يمكن نقلها بسهولة بين أجهزة الحاسوب. أن المبادئ الأساسية الكامنة وراء كلا النوعين من الأقراص هي نفسها. تطبق العديد من المفاهيم التي نستعرضها في هذه المحاضرة على جميع أجهزة التخزين الثانوية الأخرى مثل المحركات السريعة (Flash Drives).

تخزين البيانات على القرص في هيئة مِلَفّات (Files) وهي آلية لتنظيم البيانات على وسيط إلكتروني. **المِلَفّ (File)** هو مجموعة من البيانات ذات الصلة تحمل اسمًا. من وجهة نظر المستخدم فان المِلَفّ هو أصغر وحدة من البيانات يمكن خزنها على الذاكرة الثانوية. يُعد تنظيم كل شيء في مِلَفّات عرضًا موحدًا لتخزين البيانات. أن **نظام المِلَفّات (File System)** هو الشكل المنطقي الذي يوفره نظام

التشغيل بحيث يمكن للمستخدمين استعراض وإدارة البيانات كمجموعة من المِلَفّات. غالبًا ما يتم تنظيم نظام المِلَفّات في **مجلدات** (**Folders or Directories**).

المِلَفّ هو مفهوم عام. يتم إدارة أنواع مختلفة من المِلَفّات بطرق مختلفة حسب نوعها فقد يمثل المِلَفّ برنامَج (Program) أو بيانات (Data). يمكن اعتبار المِلَفّ تسلسلًا من مجموعة من البت (bits) أو البايت (Bytes) أو السطور (Lines) أو السجلات (Records) اعتمادًا على طريقة النظر إليه. كما هو الحال مع أي بيانات في الذاكرة عليك تطبيق تفسير إلى الـ بت المخزنة في مِلَفّ قبل أن يكون لها معنى. يقرر منشئ المِلَفّ كيفية تنظيم البيانات في مِلَفّ ويجب على أي مستخدمي المِلَفّ فهم هذا التنظيم.

١-٤ المِلَفّات النصية والثنائية (Text and Binary Files)

عمومًا يمكن تصنيف جميع المِلَفّات إما إلى مِلَفّات نصية (Text files) أو مِلَفّات ثنائية (Binary files). في المِلَفّ النصي يتم تنظيم مجموعة من بايتات (Bytes) البيانات كحروف من مجموعات أحرف ASCII أو Unicode. (كما تمت مناقشتها في المحاضرة الثانية) تتطلب المِلَفّات الثنائية تفسيرًا محددًا للبتات بناءً على البيانات في المِلَفّ.

أن مصطلحات "مِلَفّ نصي" و "مِلَفّ ثنائي" مضللة إلى حد ما. وكانها تلمح إلى أن المعلومات في المِلَفّ النصي لا يتم تخزينها كبيانات ثنائية. ولكننا نعلم لان جميع البيانات على الحاسوب يتم تخزينها على شكل أرقام ثنائية. أن فائدة هذه المصطلحات هي كيفية تنسيق هذه البتات: مثلا كمجموعات من ٨ أو ١٦ بت او يتم تفسيرها كحروف أو بتنسيق خاص آخر.

تمثّل بعض هذه المعلومات على هيئة حروف ورموز لكي يسهل على الإنسان قراءة المعلومات وتعديلها. مع أنّ أن المِلَفّات النصية تحتوي فقط على أحرف إلا أن هذه الأحرف يمكن أن تمثل مجموعة متنوعة من المعلومات. على سبيل المثال قد يلتزم نظام التشغيل بتخزين الكثير من بياناته كمِلَفّات نصية مثل المعلومات حول حسابات المستخدمين. كذلك يتم تخزين برنامَج مكتوب بلغة برمجة عالية المستوى (مثل بايثون او سي++) كمِلَفّ نصي يُشار إليه أحيانًا باسم مِلَفّ المصدر (source file). يمكن استخدام أي محرر نص لإنشاء (text editor) لعرض وتغيير محتوى مِلَفّ نصي بغض النظر عن نوع المعلومات التي يحتوي عليها.

بالنسبة لأنواع المعلومات الأخرى من المنطقي والفعال تمثيل البيانات عن طريق تحديد تنسيق ثنائي وتفسير محدد. لا يمكن عندئذٍ استخدام سوى البرامج التي تم إعدادها لتفسير هذا النوع من البيانات لعرضها أو تعديلها. على سبيل المثال، تخزن العديد من أنواع المِلَفّات معلومات الصور: bitmap و GIF و JPEG و TIFF، على سبيل المثال لا الحصر. كما ناقشنا في المحاضرة الثانية مع أنّ كل منها يخزن معلومات حول صورة، فإن جميع هذه المِلَفّات تخزن تلك المعلومات بطرق مختلفة. تنسيقاتها الداخلية محددة للغاية. يجب إعداد برنامَج لعرض أو تعديل نوع معين من مِلَفّات ثنائية. لهذا السبب قد لا يتمكن برنامَج يستطيع التعامل مع صورة GIF من التعامل مع صورة TIFF، والعكس صحيح.

بعض المِلَفّات التي قد تعتقد أنها نصية في الواقع ليست كذلك. على سبيل المثال ضع في اعتبارك تقريرًا تكتبه في برنامَج معالجة النصوص (word processing) مثل برنامَج مايكروسوفت وورد (MS Word) وتحفظه على القرص. يتم تخزين المستند كمِلَفّ ثنائي لأنه بالإضافة إلى الأحرف الموجودة في المستند فإنه يحتوي على معلومات حول التنسيق والأنماط والحدود والخطوط والألوان و "الإضافات" مثل الرسومات أو قصاصات فنية. يتم تخزين بعض البيانات (الحروف نفسها) كنصوص لكن المعلومات الإضافية تتطلب أن يكون لكل برنامَج معالجة نصوص تنسيق خاص به لتخزين البيانات في مِلَفّات مستنداته.

٢-٤ أنواع المِلَفّات (Files Types)

تحتوي معظم المِلَفّات سواء كانت نصية أو ثنائية على نوع معين ومحدد من المعلومات. على سبيل المثال قد يحتوي المِلَفّ على برنامَج Java أو صورة JPEG أو مقطع صوتي MP3. بينما قد تحتوي مِلَفّات أخرى على مِلَفّات أنشأتها تطبيقات معينة مثل مستند برنامَج مايكروسوفت وورد (Word) أو رسوم برنامَج مايكروسوفت فايزو (Microsoft® Visio).

يُسمى نوع المعلومات الموجودة في المستند **بنوع المِلَفّ** (**File Type**). تتعرف معظم أنظمة التشغيل على قائمة بأنواع مِلَفّات محددة.

ان الآلية الشائعة لتحديد نوع المِلَفّ هي الإشارة إلى النوع كجزء من اسم المِلَفّ. غالبًا ما يتم فصل أسماء المِلَفّات عادةً بنقطة إلى جزءين: **الاسم الرئيسي وامتداد المِلَفّ**. يشير الامتداد إلى نوع المِلَفّ. على سبيل المثال، يشير الامتداد (**`java`**) في اسم المِلَفّ (**`MyProg.java`**) إلى أنه مِلَفّ برنامَج

كود مصدر لبرنامج بلغة جافا Java. يشير الامتداد (**jpg**) في اسم المِلَفّ (**family.jpg**) إلى أنه مِلَفّ صورة من نوع JPEG.

نوع المِلَفّ	الامتداد (Extension)
المِلَفّات النصية	**txt**
المِلَفّات الصوتية	**mp3, au, wav**
مِلَفّات الصور	**gif, tiff, jpg**
مِلَفّ معالجة النصوص	**doc, docx, wp3, odt**
مِلَفّات لغات البرمجة	**java, c, cpp, py**

شكل ٢: بعض المِلَفّات الشائعة مع امتداداتها

مصدر الشكل كتاب Computer Science Illuminated by Nell B. Dale مع بعض الاضافات

أن تعريف المِلَفّات بهذه الطريقة يسهل على نظام التشغيل التعرف على نوع المِلَفّات لكي يعالجها بالطريقة المناسبة كما أنها تسهل على المستخدم التعرف عليها. يحتفظ نظام التشغيل بقائمة بأنواع المِلَفّات المعروفة ويربط كل نوع مِلَفّ بنوع معين من التطبيق. في نظام التشغيل الذي يحتوي على واجهة مستخدم رسومية (GUI) غالبًا ما يكون شكل صورة المِلَفّ (icon) مرتبط بنوع المِلَفّ أيضًا. عندما ترى مِلَفًّا في مجلد يتم عرضه بالشكل المناسب في واجهة المستخدم الرسومية مما يسهل على المستخدم تحديد نوع مِلَفّ بسرعة لأنه كل من اسم المِلَفّ وشكله يشيران إلى نوع المِلَفّ. عندما تنقر نقرًا مزدوجًا على صورة المِلَفّ لفتح البرنامَج يبدأ نظام التشغيل بتشغيل البرنامَج المرتبط بنوع المِلَفّ هذا مع تحميل للمِلَفّ.

على سبيل المثال قد تفضل برنامَج محرر (Editor) معينًا تستخدمه عند تطوير برنامَج Java. يمكنك تسجيل امتداد المِلَفّ (java) مع نظام التشغيل وربطه بذلك المحرر. ثم كلما تفتح مِلَفّ بامتداد (java) يقوم نظام التشغيل تشغيل البرنامَج المحرر المناسب. تختلف تفاصيل كيفية ربط امتداد بتطبيق برنامَج وفقًا لنظام التشغيل الذي تستخدمه.

ترتبط بعض امتدادات المِلَفّات ببرامج معينة افتراضيًا (تلقائيًا من قبل نظام التشغيل) والتي يمكنك تغييرها إن أمكن. في بعض الحالات قد يرتبط نوع المِلَفّ بأنواع مختلفة من التطبيقات لذلك لديك بعض الخيارات. على سبيل المثال قد يربط نظامك حاليًا امتداد (gif) بمستعرض ويب معين بحيث

عندما تفتح مِلَفّ صورة GIF يتم عرضه في نافذة المتصفح تلك. يمكنك اختيار تغيير الارتباط بحيث عندما تفتح مِلَفّ GIF يتم وضعه في محرر الصور المفضل لديك بدلًا من ذلك.
امتداد المِلَفّ هو مجرد دلالة على ما يحتوي عليه المِلَفّ. يمكنك تسمية مِلَفّ أي شيء تريده (طالمًا تستخدم الأحرف التي يسمح بها نظام التشغيل لأسماء المِلَفّات). يمكنك منح أي مِلَفّ امتداد .gif، على سبيل المثال، لكن هذا لا يجعله مِلَفّ صورة GIF. لا يؤدي تغيير الامتداد إلى تغيير البيانات في المِلَفّ أو تنسيقه الداخلي. إذا حاولت فتح مِلَفّ مسيء التسمية في برنامج يتوقع تنسيقًا معينًا ستتلقى رسالة خطأ.

٣-٤ عمليات المِلَفّات (File Operations)

بمساعدة نظام التشغيل يمكنك إجراء أي من العمليات العديدة التالية على مِلَفّات:

- **إنشاء مِلَفّ**: يخصص نظام التشغيل مساحة كافية في نظام المِلَفّات للمحتوى الجديد ويضع إدخالًا للمِلَفّ في جدول الدليل المناسب ويسجل اسم وموقع المِلَفّ.
- **حذف مِلَفّ**: يحدد نظام التشغيل أن مساحة الذاكرة التي كان يستخدمها المِلَفّ سابقًا أصبحت الآن خالية ويزيل الإدخال المناسب في جدول الدليل.
- **فتح مِلَفّ**: يتطلب معظم أنظمة التشغيل فتح المِلَفّ قبل إجراء عمليات القراءة والكتابة عليه. يحتفظ نظام التشغيل بجدول صغير لجميع المِلَفّات المفتوحة حاليًا لتجنب الاضطرار إلى البحث عن المِلَفّ في نظام المِلَفّات الكبير في كل مرة يتم فيها إجراء عملية لاحقة. لإغلاق المِلَفّ عندما لم يعد قيد الاستخدام النشط، يزيل نظام التشغيل الإدخال في جدول المِلَفّات المفتوحة.
- **قراءة البيانات من مِلَفّ**: يعني ذلك أن نظام التشغيل يقدم نسخة من البيانات الموجودة في المِلَفّ، بدءًا من مؤشر المِلَفّ الحالي. بعد القراءة، يتم تحديث مؤشر المِلَفّ.
- **كتابة البيانات إلى مِلَفّ**: يتم تخزين البيانات في الموقع الذي يشير إليه مؤشر المِلَفّ الحالي، ثم يتم تحديث مؤشر المِلَفّ. غالبًا ما يسمح نظام التشغيل بفتح مِلَفّ للقراءة أو الكتابة، ولكن ليس لكلا العمليتين في نفس الوقت.

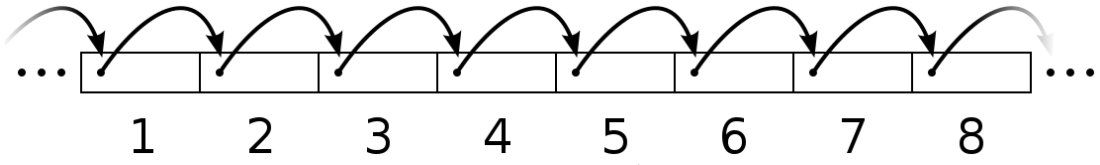
- **إعادة تموضع مؤشر المِلَفّ الحالي**: قد يتم إعادة تموضع مؤشر المِلَفّ الحالي لمِلَفّ مفتوح إلى موقع آخر في المِلَفّ للتحضير لعملية القراءة أو الكتابة التالية. يتطلب إلحاق البيانات بنهاية مِلَفّ تحديد مؤشر المِلَفّ إلى نهاية المِلَفّ، ثم يتم كتابة البيانات المناسبة في ذلك الموقع.
- **قص (اختصار) مِلَفّ**: يعني حذف محتويات المِلَفّ دون إزالة المدخلات الإدارية في جداول المِلَفّات. تتجنب هذه العملية الحاجة إلى حذف مِلَفّ ثم إعادة إنشائه. في بعض الأحيان، تكون عملية القص متطورة بما يكفي لحذف جزء من مِلَفّ، من مؤشر المِلَفّ الحالي إلى نهاية المِلَفّ.
- **إعادة تسمية مِلَفّ**: يوفر نظام التشغيل أيضًا عملية لتغيير اسم مِلَفّ، والتي تسمى إعادة تسمية المِلَفّ. وبالمثل، يوفر القدرة على إنشاء نسخة كاملة من محتويات مِلَفّ، وإعطاء النسخة اسمًا جديدًا.

٤.٤ الوصول إلى المِلَفّات (File Access)

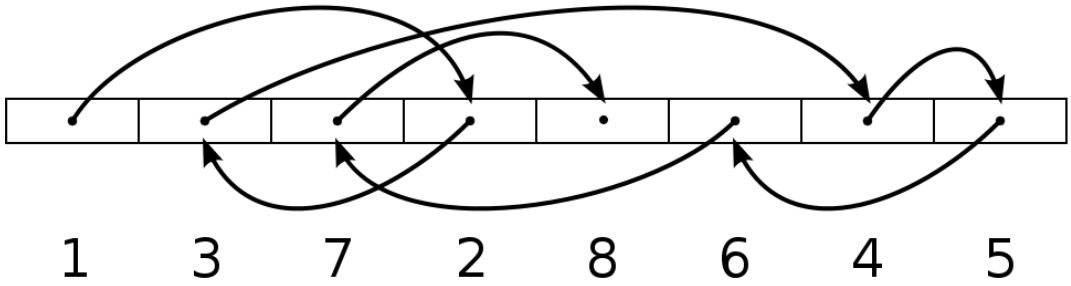
يمكن الوصول إلى البيانات الموجودة في مِلَفّ بعدة طرق مختلفة. توفر بعض أنظمة التشغيل نوعًا واحدًا فقط من الوصول إلى المِلَفّات في حين يوفر البعض الآخر مجموعة متنوعة من طرق الوصول. يتم تحديد نوع الوصول لمِلَفّ معين عند إنشائه.

هناك تقنيتين للوصول إلى المِلَفّات: الوصول التسلسلي (sequential access) والوصول المباشر (direct access). تتشابه الاختلافات بين هاتين التقنيتين مع الاختلافات بين الطبيعة التسلسلية للشريط المغناطيسي والوصول المباشر الذي يوفره القرص المغناطيسي، كما تمت مناقشتها في المحاضرة الثالثة. ومع ذلك يمكن تخزين كلا النوعين من المِلَفّات على أي نوع من الوسائط. تحدد تقنيات الوصول إلى المِلَفّات الطرق التي يمكن بها إعادة تحديد موضع مؤشر المِلَفّ الحالي. وهي مستقلة عن القيود المادية للأجهزة التي يتم تخزين المِلَفّ عليها.

أكثر تقنيات الوصول شيوعًا، وأبسطها في التنفيذ، هو **الوصول التسلسلي للمِلَفّات** (**sequential file access**) والذي يرى المِلَفّ كبنية خطية. يتطلب الأمر معالجة البيانات في المِلَفّ بالترتيب. تحرك عمليات القراءة والكتابة مؤشر المِلَفّ الحالي وفقًا لحجم البيانات التي تتم قراءتها أو كتابتها. تسمح بعض الأنظمة بإعادة ضبط مؤشر المِلَفّ إلى بداية المِلَفّ و/أو التخطي للأمام أو للخلف بعدد معين من السجلات (شكل ٣ ـ ا).



ا.الوصول التسلسلي للمِلَفّات (sequential file access)



ب. الوصول المباشر للمِلَفّات (direct file access)

شكل ٣: تقنيات الوصول إلى المِلَفّات

مصدر الشكل موقع https://www.itrelease.com

تقسم **الوصول المباشر للمِلَفّات (direct file access)** إلى سجلات منطقية مرقمة. يسمح الوصول المباشر للمستخدم بتعيين مؤشر المِلَفّ إلى أي سجل معين عن طريق تحديد رَقَم السجل. لذا يمكن للمستخدم قراءة السجلات وكتابتها بأي ترتيب خاص يريده كما هو موضح في الشكل ٣ ـ ب. تعد مِلَفّات الوصول المباشر أكثر تعقيدًا في التنفيذ ولكنها مفيدة في المواقف التي يجب فيها توفير أجزاء محددة من مجموعات البيانات الكبيرة بسرعة مثل قاعدة البيانات (Database).

٥ـ٤ حماية المِلَفّات (File Protection)

في أنظمة التشغيل متعددة المستخدمين تعدّ حماية المِلَفّات ذات أهمية قصوى. لا نريد أن يتمكن مستخدم واحد من الوصول إلى مِلَفّات مستخدم آخر ما لم يُسمح بهذا الوصول بشكل صريح. تقع على عاتق نظام التشغيل مسؤولية ضمان الوصول الصالح إلى المِلَفّات. تقوم أنظمة التشغيل المختلفة بحماية المِلَفّات بطرق مختلفة. عمومًا تحدد آلية حماية المِلَفّات من يمكنه استخدام مِلَفّ ولأي غرض.

على سبيل المثال يتم تقسيم إعدادات حماية المِلَفّ في نظام التشغيل UNIX إلى ثلاث فئات: المالك (Owner) والمجموعة (Group) والعالم (World). تحت كل فئة يمكنك تحديد ما إذا كان يمكن قراءة (read) المِلَفّ أو كتابته (written) أو تنفيذه (executed). بموجب هذه الآلية إذا كان بإمكانك الكتابة إلى مِلَفّ يمكنك أيضًا حذفه.

كل مِلَفّ "مملوك" من قبل مستخدم معين والذي غالبًا ما يكون هو منشئ المِلَفّ. عادةً ما يتمتع المالك بأقوى الصلاحيات فيما يتعلق بالمِلَفّ. قد يرتبط المِلَفّ أيضا بمجموعة والتي هي ببساطة قائمة بالمستخدمين. تنطبق أذونات المجموعة على جميع المستخدمين في المجموعة المرتبطة. قد تقوم بإنشاء أذونات للمجموعة على سبيل المثال لجميع المستخدمين الذين يعملون على مشروع معين. أخيرًا تنطبق أذونات العالم على أي شخص لديه حق الوصول إلى نظام التشغيل. نظرًا لأن هذه الأذونات تمنح حق الوصول إلى أكبر عدد من المستخدمين فإنها عادةً ما تكون الأكثر تقييدًا.

باستخدام هذه التقنية، يمكن عرض الأذونات كما في الجدول التالي:

	قراءة (Read)	كتابة (Write)/حذف (Delete)	تنفيذ (Execute)
المالك (Owner)	نعم	نعم	لا
المجموعة (Group)	نعم	لا	لا
العالم (World)	لا	لا	لا

جدول ١: اذونات المِلَفّات حسب الفئات

مصدر الشكل كتاب Computer Science Illuminated by Nell B. Dale

لنفترض أن هذه الشبكة تمثل الصلاحيات على مِلَفّ بيانات يُستخدم في مشروع يدعى ألفا. يمكن لمالك المِلَفّ (ربما مدير المشروع) القراءة منه أو الكتابة فيه. لنفترض أيضًا أن المالك يقوم (باستخدام نظام التشغيل) بإعداد مجموعة تسمى فريق ألفا والتي تضم جميع أعضاء فريق المشروع ويربط هذه المجموعة بمِلَفّ البيانات هذا. يمكن لأعضاء فريق ألفا قراءة البيانات في المِلَفّ ولكن لا يمكنهم تغييرها. لا يُمنح أي شخص آخر أي إذن للوصول إلى المِلَفّ. لاحظ أنه لم يُمنح أي مستخدم امتيازات تنفيذ للمِلَفّ لأنه مِلَفّ بيانات وليس برنامجًا قابلًا للتنفيذ.

تضع أنظمة تشغيل أخرى مخططات حماية لها بطرق مختلفة لكن الهدف هو نفسه: التحكم في الوصول لحماية نفسنا من المحاولات المتعمدة للوصول غير الملائم وكذلك تقليل المشاكل غير المقصودة التي يسببها المستخدمون ذوو النيات الحسنة ولكن شديد الخَطَر.

٥ الأدلة (Directories)

كما هو مذكورا علاه فان الدليل يضم مجموعة من المِلَفّات وله اسم محدد. إنها طريقة لتجميع المِلَفّات بحيث يمكنك تنظيمها بطريقة منطقية. على سبيل المثال قد تضع جميع أوراقك وملاحظاتك لمحاضرة

معينة في دليل تم إنشاؤه لتلك المحاضرة. يجب على نظام التشغيل تتبع الأدلة والمِلَفّات التي تحتويها بعناية.

في معظم أنظمة التشغيل يمثّل الدليل بمِلَفّ. يحتوي مِلَفّ الدليل على بيانات حول المِلَفّات الأخرى في الدليل. بالنسبة لأي مِلَفّ معين، يحتوي الدليل على اسم المِلَفّ ونوع المِلَفّ والعنوان على القرص حيث يتم تخزين المِلَفّ والحجم الحالي للمِلَفّ. كما يحتوي الدليل أيضًا على معلومات حول أعدادات الحماية الموضوعة على المِلَفّ. بالإضافة إلى ذلك قد يحتوي على معلومات تصف وقت إنشاء المِلَفّ وآخر تعديل عليه.

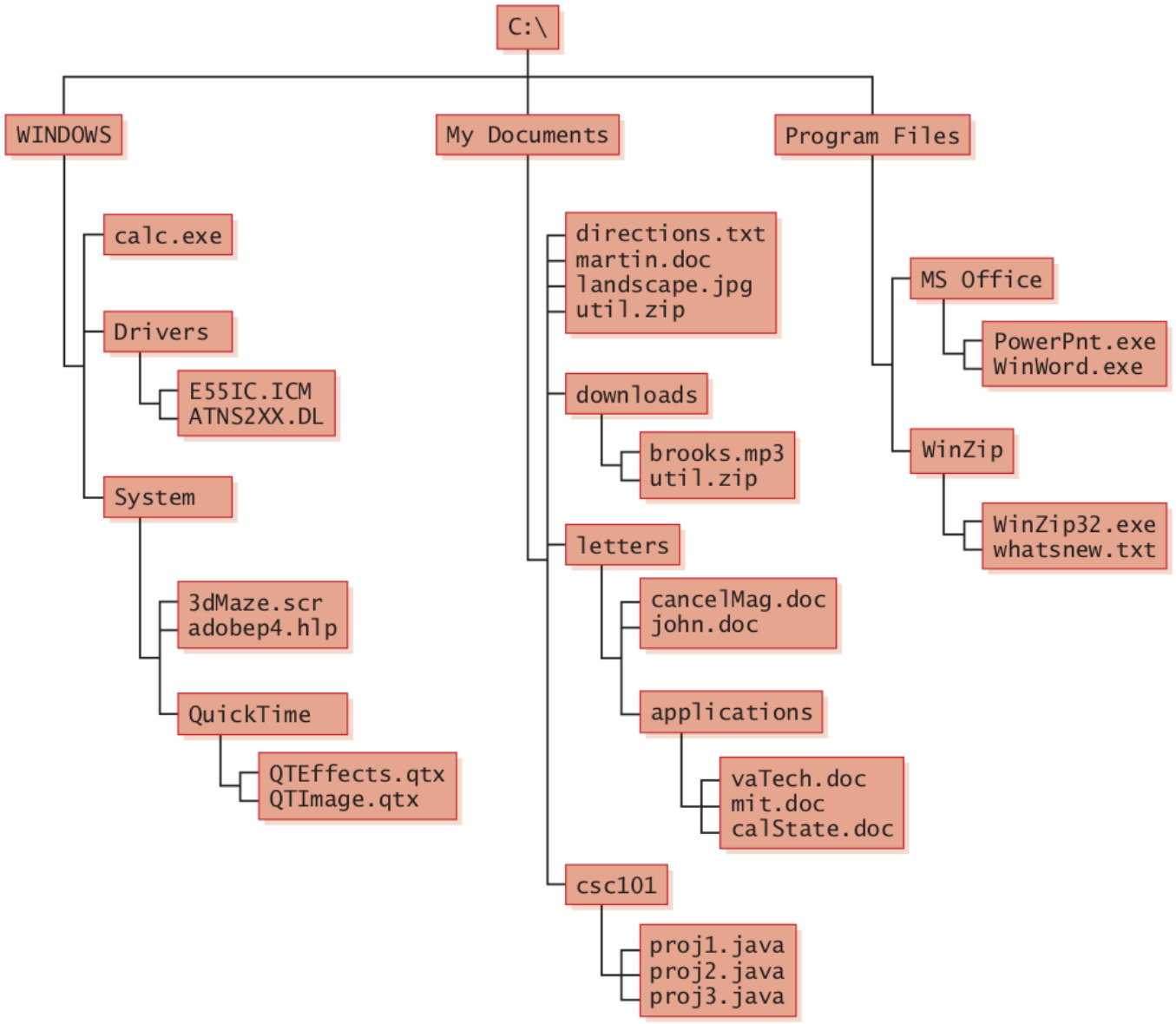
يجب أن يكون مِلَفّ الدليل قادرًا على دعم العمليات الشائعة التي يتم إجراؤها على مِلَفّات الأدلة. على سبيل المثال يجب أن يكون المستخدم قادرًا على استعراض جميع المِلَفّات في الدليل. كذلك العمليات الشائعة الأخرى مثل إنشاء وحذف وإعادة تسمية المِلَفّات داخل الدليل. علاوة على ذلك غالبًا ما يتم البحث في الدليل لمعرفة ما إذا كان مِلَفّ معين موجودًا في الدليل.

مسألة رئيسية أخرى عند إدارة الأدلة وهي الحاجة إلى طبيعة العلاقات بين الأدلة وهو ما تم مناقشته في القسم التالي.

٥.١ أشجار الأدلة (Directory Trees)

يمكن أن يكون دليل من المِلَفّات داخل دليل آخر. عادةً ما يُسمى الدليل الذي يحتوي على دليل آخر **بالدليل الرئيسي** (**parent directory**) ويُسمى الدليل الموجود بالداخل **بالدليل الفرعي** (**subdirectory**). يمكنك إعداد مثل هذه الأدلة المتداخلة حسب الحاجة للمساعدة في تنظيم نظام المِلَفّات. يمكن أن يحتوي دليل واحد على العديد من الأدلة الفرعية. كذلك يمكن أن تحتوي الأدلة الفرعية على أدلة فرعية خاصة بها مما يخلق هيكلًا هرميًا. لتوضيح هذا التسلسل الهرمي غالبًا ما يُنظر إلى نظام المِلَفّات على أنه شجرة من الأدلة (directory tree) تُظهر الأدلة والمِلَفّات داخل الأدلة الأخرى. يُسمى الدليل على أعلى مستوى بجذر الدليل (root directory).

كمثال لاحظ شجرة الدليل الموضحة في الشكل ٤. تمثل هذه الشجرة جزءًا صغيرًا جدًا من نظام المِلَفّات الذي قد يوجد على جهاز حاسوب يستخدم نوعًا من نظام التشغيل Microsoft Windows. يُشار إلى جذر نظام الدليل باستخدام حرف محرك الأقراص C: متبوعًا بشرطة مائلة (\).



شكل ٤: مثال لشجرة أدلة في نظام وندوز (Windows)

مصدر الشكل كتاب Computer Science Illuminated by Nell B. Dale

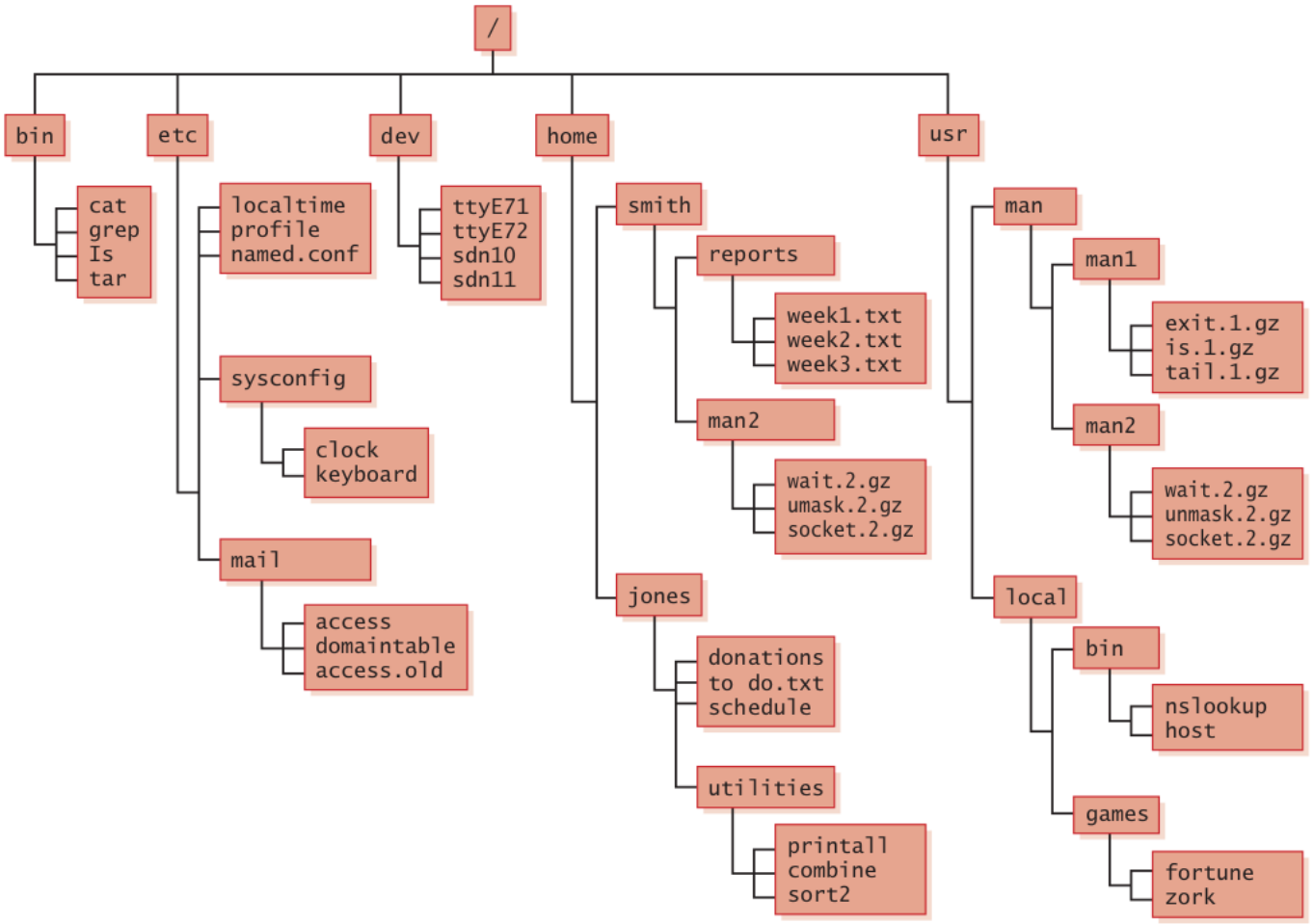
في شجرة الدليل هـــذه يحتـــوي الدليل الرئيسي على ثلاثـــة أدلة فرعيـــة: WINDOWS و My Documents و Program Files. يوجد داخل دليـل WINDOWS مِلَـفّ يسـمى calc.exe بالإضافة إلى دليلان فرعيان آخران (Drivers و System). تحتوي تلك الأدلة على مِلَفّات أخرى وأدلة فرعية. لاحظ أن جميع هذه الأدلة في نظام حقيقي ستحتوي عادةً على عدد أكبر بكثير من الأدلة الفرعية والمِلَفّات.

غالبًا ما تستخدم الحواسيب الشخصية تشبيهًا للمجلدات لتمثيل هيكل الدليل مما يعزز فكرة الاحتواء (المجلدات داخل المجلدات الأخرى مع احتواء بعض المجلدات في النهاية على مستندات أو بيانات

أخرى). غالبًا ما يكون الرمز المستخدم لإظهار دليل في الواجهة الرسومية لنظام التشغيل صورة لمجلد مِلَفّات من ورق بني اللون مثل النوع الذي نستخدمة في الواقع.

في الشكل ٤، يوجد مِلَفّان بنفس الاسم util.zip (الواحد في مجلد MyDocuments والثاني في مجلد فرعي له يسمى downloads). تسمح بنية الدليل المتداخلة لوجود أسماء متشابهة لعدة مِلَفّات. يجب أن يكون لجميع المِلَفّات داخل نفس الدليل أسماء فريدة (غير متشابهه) ولكن يمكن للمِلَفّات في أدلة أو متفرعات مختلفة أن تملك نفس الاسم. قد تحتوي هذه المِلَفّات أو لا تحتوي على نفس البيانات الشيء الوحيد الذي نعرفه هو أنها متطابقة في الاسم.

في أي لحظة، يمكنك اعتبار نفسك تعمل في موقع معين (أي دليل فرعي معين) لنظام المِلَفّات. يُشار إلى هذا الدليل الفرعي باسم **الدليل العامِل** الحالي (**working directory**). كلما "تحركت" داخل نظام المِلَفّات يتغير الدليل العامِل الحالي.



شكل ٥: مثال لشجرة أدلة في نظام يونكس (UNIX)

مصدر الشكل كتاب Computer Science Illuminated by Nell B. Dale

تمثل شجرة الدليل في الشكل ٥ نموذجًا لنظام مِلَفّات يونيكس (UNIX). بالمقارنة مع شجرة الدليل في الشكل ٤ والتي تمثل نموذجًا لنظام مِلَفّات نظام التشغيل وندوز (Windows) نجد أن في كلًا الشكلين يطبق مفهوم احتواء الأدلة الفرعية مع اختلاف قواعد تسمية المِلَفّات والأدلة. لقد تم تطوير نظام يونيكس كبيئة للبرمجة والعمل على مستوى النظام لذلك فهو يستخدم أسماء مختصرة وغامضة أكثر للمِلَفّات والأدلة. أيضًا يُشار في بيئة يونيكس إلى الجذر باستخدام شرطة مائلة للأمام (/).

٢.٥ اسماء المسار (Path Names)

كيف نحدد مِلَفًا أو دليلًا فرعيًا معينًا؟ هناك عدة طرق للقيام بذلك. إذا كنت تعمل في **واجهة مستخدم رسومية** (GUI) لنظام التشغيل فيمكنك النقر نقرًا مزدوجًا بالفأرة (Mouse) لفتح الدليل والاطلاع على محتوياته. تعرض نافذة الدليل النشط محتويات دليل العمل الحالي. يمكنك الاستمرار في "التنقل" عبر نظام المِلَفّات باستخدام نقرات الفأرة لكي تغير دليل العمل الحالي حتى تجد المِلَفّ أو الدليل المطلوب. لتحريك بنية الدليل لأعلى يمكنك غالبًا استخدام أيقونة على شريط النافذة أو خيار قائمة منبثقة للانتقال إلى الدليل الأصلي.

توفر معظم أنظمة التشغيل أيضًا **واجهة غير رسومية** (قائمة على النص) لنظام التشغيل. لذلك يجب أن نكون قادرين على تحديد مواقع المِلَفّات باستخدام النص. تعدّ هذه القدرة مهمة جدًا لتعليمات النظام المخزنة في مِلَفّات الأوامر التنفيذية (Batch command files) لنظام التشغيل. يمكن استخدام أوامر مثل **cd** (والتي تعني تغيير الدليل Change Directory) في الوضع النصي لتغيير دليل العمل الحالي.

للإشارة إلى مِلَفّ معين باستخدام النص نحدد مسار ذلك المِلَفّ وهو سلسلة من الأدلة التي يجب أن تذهب من خلالها للعثور على المِلَفّ. قد يكون المسار مطلقًا أو نسبيًا. يبدأ اسم المسار المطلق من الجذر ويحدد كل خطوة أسفل الشجرة حتى يصل إلى المِلَفّ أو الدليل المطلوب. يبدأ اسم المسار النسبي من دليل العمل الحالي.

دعونا نلقي نَظْرَة على أمثلة لكل نوع من المسارات. فيما يلي أسماء المسارات المطلقة بناءً على شجرة الدليل الموضحة في الشكل ٤:

```
C:\Program Files\MS Office\WinWord.exe
C:\My Documents\letters\applications\vaTech.doc
C:\Windows\System\QuickTime
```

يبدأ كل اسم مسار من الجذر ويستمر إلى أسفل بنية الدليل. يتم فصل كل دليل فرعي بالشرطة المائلة العكسية (\). لاحظ أن المسار يمكن أن يحدد مستندًا محددًا (كما هو الحال في المثالين الأولين) أو دليلًا فرعيًا بأكمله (كما هو الحال في المثال الثالث).

تعمل المسارات المطلقة في نظام UNIX بنفس الطريقة باستثناء أن الحرف المستخدم لفصل الدلائل الفرعية هو الشرطة المائلة للأمام (/). فيما يلي بعض الأمثلة على أسماء المسارات المطلقة التي تتوافق مع شجرة الدليل في الشكل ٥:

```
/bin/tar
/etc/sysconfig/clock
/usr/local/games/fortune
/home/smith/reports/week1.txt
```

تعتمد أسماء المسارات النسبية على دليل العمل الحالي. أي أنها مرتبطة بموضعك الحالي. لنفترض أن دليل العمل الحالي هو **C:\My Documents\letters** (من الشكل ٤).

اذن يمكن استخدام أسماء المسارات النسبية التالية:

```
cancelMag.doc
applications\calState.doc
```

يحدد المثال الأول اسم المِلَفّ فقط والذي يمكن العثور عليه في دليل العمل الحالي. يحدد المثال الثاني مِلَفًّا في الدليل الفرعي للتطبيقات. بحكم التعريف يقع الجزء الأول من أي مسار نسبي صالح في دليل العمل.

في بعض الأحيان، عند استخدام مسار نسبي نحتاج إلى العمل في طريقنا إلى أعلى الشجرة. لاحظ أن هذا الاعتبار لم يكن مشكلة عندما استخدمنا المسارات المطلقة. في معظم أنظمة التشغيل يتم استخدام نقطتين (..) لتحديد الدليل الأصلي (يتم استخدام نقطة واحدة لتحديد دليل العمل الحالي). لذلك إذا كان دليل العمل هو **C:\My Documents\letters** فإن أسماء المسارات النسبية التالية تكون صالحة أيضًا:

```
..\landscape.jpg
..\csc111\proj2.java
..\..\WINDOWS\Drivers\E55IC.ICM
..\..\Program Files\WinZip
```

تعمل أنظمة UNIX بشكل أساسي بنفس الطريقة. باستخدام شجرة الدليل في الشكل ٥ وبافتراض أن دليل العمل الحالي هو **home/jones/** فإن ما يلي هو أسماء مسارات نسبية صالحة:

```
utilities/combine
../smith/reports
../../dev/ttyE71
../../usr/man/man1/ls.1.gz
```

تسمح معظم أنظمة التشغيل للمستخدم بتحديد مجموعة من المسارات التي يتم البحث فيها (بترتيب معين) للمساعدة في حل الإشارات إلى البرامج القابلة للتنفيذ. غالبًا ما يتم تحديد مجموعة المسارات هذه باستخدام متغير نظام التشغيل المسمى PATH والذي يحتوي على سلسلة تحتوي على عدة أسماء مسارات مطلقة. لنفترض، على سبيل المثال، أن المستخدم jones (من الشكل ٥) لديه مجموعة من البرامج المساعدة التي يستخدمها من وقت لآخر. يتم تخزينها في الدليل **/home/jones/utilities**. عند إضافة هذا المسار إلى متغير PATH فإنه يصبح موقعًا قياسيًا يستخدم للعثور على البرامج التي يحاول المستخدم jones تنفيذها. وبغض النظر عن دليل العمل الحالي عندما يقوم جونز بتنفيذ برنامج printall (الاسم فقط) سوف يتم العثور عليه في دليل الأدوات المساعدة الخاص به.