بناء توزيعتك الخاصة:

دليل شامل لـ

ARCHISO، CALAMARES، و HELWAN LINUX

عن المؤلف

اسمي سعيد محمد عبد الملك، وعرفت باسم **"سعيد بدر الدين"**، إلا أن الكثيرين في مجتمع التقنية يعرفونني باسم **"S.M.A"**، وهو اختصار لقناتي على يوتيوب **SMA Coding**. على الرغم من أن خلفيتي الأكاديمية تكمن في المحاسبة المالية، إلا أن شغفي الحقيقي كان دائمًا يكمن في التكنولوجيا. هذا الشغف دفعني باستمرار لاستكشاف عالم الحوسبة والبرمجيات الحرة، ودائمًا ما كنت أسأل نفسي: **"**كيف يعمل هذا؟**"** و**"**هل يمكنني بناء شيء أفضل؟**."**

هذا الفضول المستمر قادني عميقًا إلى عالم أنظمة التشغيل، حيث اكتشفت فلسفة **آرش لينكس** —فلسفة تتماشى تمامًا مع رؤيتي الخاصة: البساطة، والتحكم الكامل، والقدرة على بناء أي شيء من الصفر. من هذا المبدأ، ولدت فكرة **Helwan Linux**. لم يكن مجرد مشروع تقني، بل كان حلمًا بإنشاء توزيعة تعكس هوية المجتمع العربي، وتقدم تجربة مستخدم سلسة وشخصية، وتقف كمثال ملهم لما يمكن تحقيقه بالشغف والمعرفة.

على مر السنين، قمت بتوثيق هذه الرحلة على قناتيSMA Coding ، ساعيًا لمشاركة كل ما تعلمته مع المبرمجين وعشاق لينكس في كل مكان. هذا الكتاب هو تتويج لتلك الرحلة: ليس مجرد دليل تقني، بل هو خلاصة سنوات من التجربة والخطأ، والتعلم، والمشاركة. آمل أن تجدوا في هذه الصفحات كل ما تحتاجونه لتحويل أفكاركم إلى واقع — وأن تلهمكم لتلعبوا دورًا فاعلاً في مجتمع المصادر المفتوحة.

إهداء

ثمن هذا الكتاب، في أي مكان في العالم، هو أن تقدم دعاءً خالصاً بالرحمة لوالديّ، بغض النظر عن معتقداتك .

التكلفة الحقيقية لهذا الكتاب هي أن تشاركه على أوسع نطاق ممكن.

إهداء

إلى والدي الحبيب،

إلى والدتي الحبيبة،

إلى أخواتي، نبض قلبي.

إلى أحبائي :

نور عيني، ابني محمد؛

بهجة قلبي، ابني عمر؛

وابنتي الغالية ملك .

أنتم أعظم إلهام في حياتي، وحضوركم يذكرني باستمرار بأهمية بناء شيء يستحق أن يُورّث .

هذا الكتاب لكم، عربوناً على أن الشغف والمعرفة يمكن أن يخلقا عالماً أفضل.

إهداء

إلى عمالقة التكنولوجيا الذين شكلوا عالمنا الرقمي:

- **دينيس ريتشي وكين تومبسون**، الأبوان الروحيان للغة C ونظامUnix ، اللذان وضعا الأساس لكل ما نستخدمه اليوم.
- **ريتشارد ستالمان**، فيلسوف البرمجيات الحرة، الذي زرع بذور الحركة.
- **لينوس تورفالدس**، الذي جعل لينكس حقيقة ووحدنا تحت راية واحدة.
- وإلى كل مطور ومبرمج يعمل خلف الكواليس، ويسهر ليالٍ طويلة لإصلاح الأخطاء وكتابة الأكواد.

إلى أيقونات هذا المجتمع:

- **النينجا :(The Ninjas)** المطورون الذين يختبئون خلف شاشة سوداء، ويكتبون الأكواد بصمت وبإتقان.
- **الهاكرز :(Hackers)** ليسوا مدمرين، بل مستكشفين للأنظمة، يتعقبون نقاط الضعف للفهم والتحسين.
- **المساهمون :(Contributors)** أولئك الذين يضيفون قطعاً صغيرة تصبح اللبنات الأساسية للمشاريع الضخمة، من سطر كود واحد إلى ترجمة وثيقة.
- **مستخدمو آرش لينكس :(Arch Linux users)** الذين يجسدون فلسفة "افعلها بنفسك (DIY) "في حياتهم ويؤمنون بالتحكم الكامل في أنظمتهم.

هذا الإهداء هو عربون امتنان لمجتمع يؤمن بالعطاء بلا حدود .فلتستمر شعلة المعرفة في الاحتراق ببراعة.

عن الكتاب: خارطة طريق للإبداع الرقمي

هل تخيلت يومًا أن نظام تشغيل لينكس يمكن تصميمه خصيصًا ليناسب ذوقك واحتياجاتك الفريدة؟ هل حلمت بامتلاك توزيعة تحمل بصمتك الشخصية، وتحتوي على الأدوات التي تستخدمها يوميًا، وتلبي احتياجات مجتمعك المحلي أو عالمك المهني؟

إذا كانت هذه الأفكار قد راودتك، فإن هذا الكتاب ليس مجرد دليل، بل هو رفيقك في رحلة تحويل هذا الحلم إلى واقع ملموس. إنه دعوة مفتوحة لكل مبدع، مطور، وعاشق لأنظمة التشغيل للانتقال من عالم الاستهلاك إلى عالم الإنتاج. معًا، سننطلق في رحلة مثيرة تبدأ بالجذور العميقة لنظام لينكس، مرورًا باستكشاف الأسرار التقنية للأدوات القوية التي يستخدمها المحترفون، وصولًا إلى بناء مشروعك الخاص من الصفر.

ماذا ستجد في صفحات هذا الكتاب؟

يتجاوز هذا الكتاب كونه مجرد مجموعة من التعليمات التقنية؛ إنه خارطة طريق استراتيجية تمكنك من الغوص في أعماق عالم لينكس المفتوح، وتمنحك أدوات للتحكم الكامل. مع كل فصل، ستكتسب مهارات متقدمة تضعك في مقعد القيادة:

- **فهم عميق وفلسفي لآرش لينكس :(Arch Linux)** لن تتعلم فقط كيفية استخدام آرش، بل ستفهم الفلسفة التي بني عليها. ستكتشف مبدأ "التحكم الكامل" وكيف يمكن لهذا الأساس المتين أن يكون لوحة قماشية فارغة لمشروعك، مما يحررك من القيود التي تفرضها التوزيعات الأخرى.
- **إتقان Archiso من الصفر إلى الاحتراف :**هذا هو جوهر الكتاب. ستتحول من مستخدم عادي إلى "مهندس إصدارات". ستتعلم كيفية بناء صور ISO قابلة للإقلاع وتخصيص كل جانب من جوانب النظام المباشر بدقة، بما في ذلك إضافة الحزم، واختيار بيئات سطح المكتب، وكتابة نصوص برمجية مخصصة لأتمتة مهامك.
- **تحكم كامل في :Calamares** ستفهم كيفية دمج واجهة تثبيت رسومية سلسة واحترافية في توزيعتك. ستتعمق في تخصيص كل تفصيل فيCalamares ، من تصميم واجهة المستخدم إلى تعديل وحدات التثبيت لضمان تجربة تثبيت فريدة تتماشى تمامًا مع رؤيتك.
- **تحويل الأفكار إلى واقع: بناء توزيعتك الخاصة :**بعد إتقان الأدوات، ستمتلك المعرفة والمهارات اللازمة لتحويل أفكارك المجردة إلى توزيعة لينكس عملية، مستقرة، وجاهزة للاستخدام. ستتجاوز مجرد إضافة الحزم لتصل إلى مرحلة بناء نظام شامل ومترابط.
- **الاستلهام من تجربة حلوان لينكس :(Helwan Linux)** سنقدم لك نموذجًا عمليًا وملهمًا من قلب المجتمع العربي. تجربة حلوان لينكس هي شهادة حية على أن الشغف والمعرفة يمكن أن ينتجا مشروعًا قويًا ومفيدًا. من خلال مراجعة هذا المشروع، ستكتسب رؤى عملية حول كيفية تجميع كل القطع معًا، وكيف يمكن لمجتمع صغير بناء شيء ذي تأثير كبير.

لمن هذا الدليل؟

لا يقتصر هذا الكتاب على فئة معينة، بل تم تصميمه بعناية ليكون بمثابة جسر يربط بين مختلف مستويات الخبرة في عالم لينكس:

- **للمبتدئ الطموح :**إذا كنت قد بدأت رحلتك للتو في عالم لينكس، فسيرشدك هذا الكتاب خطوة بخطوة. سنبدأ بشرح المفاهيم المعقدة بأسلوب مبسط، مع التركيز على أمثلة عملية تمكنك من بناء مشروعك الأول بنجاح.
- **للمستخدم المتوسط :**إذا كنت على دراية بأساسيات لينكس وتستخدم توزيعات مثل آرتش أو غيرها، فسيفتح لك هذا الكتاب آفاقًا جديدة للتحكم والتخصيص. ستنتقل من مجرد استخدام النظام إلى فهم كيفية بنائه وتعديله من الداخل.
- **للمطور المتقدم :**إذا كنت تتطلع إلى إنشاء بيئات تطوير جاهزة، أو بناء أدوات مخصصة لأغراض أمنية أو علمية، أو حتى إطلاق توزيعة لينكس لمشروعك التجاري أو المجتمعي، فستجد هنا الموارد والأمثلة المتقدمة التي تحتاجها لتنفيذ أفكارك بأعلى كفاءة.

آرتش لينكس: حجر الزاوية للمشاريع الطموحة

لم يتم اختيار آرتش لينكس كقاعدة لهذا الكتاب بشكل عشوائي؛ بل هو قرار استراتيجي يعتمد على عدة مبادئ أساسية جعلته الخيار الأمثل للمشاريع المخصصة:

- **مبدأ)KISS اجعله بسيطًا، أيها الغبي :Keep It Simple, Stupid -** يتبنى آرتش مبدأ "البساطة". هذا يعني أنه يتجنب التعقيد غير الضروري، مما يجعله شفافًا وسهل الفهم لكيفية عمله من البداية إلى النهاية. أنت تبني نظامك بنفسك وتتحكم في كل جزء منه.
- **التحكم الكامل في التخصيص :**على عكس التوزيعات التي تأتي ببيئات سطح مكتب وبرامج محددة مسبقًا، يمنحك آرتش حرية مطلقة. أنت تقرر ما هو موجود في نظامك، مما يتيح لك بناء نظام خفيف وفعال ومرن يلبي احتياجاتك بدقة.
- **نموذج الإصدار المتجدد :(Rolling Release)** تحصل على أحدث إصدارات البرامج على الفور، مما يضمن أن تكون توزيعتك دائمًا محدثة وتستفيد من أحدث الميزات وإصلاحات الأمان فور توفرها.
- **وثائق لا مثيل لها ومجتمع تعاوني :**لا يوجد نظام لينكس آخر يمتلك وثائق رسمية (Arch Wiki) بهذا المستوى من التفصيل والدقة. إلى جانب مجتمع واسع وتعاوني، ستجد دائمًا المساعدة والمعلومات التي تحتاجها للتغلب على أي تحدٍ.

حلوان لينكس: أكثر من مجرد توزيعة

في قلب هذا الدليل، لا نقدم لك الأدوات فحسب؛ بل نلهمك من خلال مثال حي وعملي .**حلوان لينكس** هي تجربة رائدة لمشروع عربي طموح مبني على فلسفة آرتش لينكس، يهدف إلى توفير تجربة لينكس سلسة ومخصصة للمستخدمين في المنطقة العربية. من خلال تحليل هذا المشروع، ستتعلم كيفية تطبيق المفاهيم التي سنتعلمها، وكيف يمكن للشغف الجماعي أن يخلق شيئًا ذا تأثير حقيقي. حلوان لينكس ليست مجرد توزيعة؛ إنها شهادة حية على أن الإبداع لا يعرف حدودًا وأن المعرفة يمكن تحويلها إلى مشاريع مجتمعية قوية ومفيدة.

نصائح للحصول على أقصى استفادة من هذه الرحلة

المعرفة النظرية وحدها لا تكفي. للحصول على أقصى استفادة من هذا الكتاب، ندعوك لتبني عقلية الباحث والمجرب:

- **التطبيق العملي هو مفتاحك :**لا تقرأ فقط. قم بإنشاء بيئة افتراضية وابدأ في تطبيق كل خطوة؛ التجربة العملية هي أفضل معلم.
- **آرتش ويكي هو أفضل صديق لك :**لا تتردد في الرجوع إلى آرتش ويكي للحصول على تفاصيل إضافية حول أي حزمة أو أمر.
- **كن فضوليًا ومغامرًا :**قم بتغيير الإعدادات، وأضف برامج جديدة، وحاول فهم ما يحدث خلف الكواليس.
- **المشاركة المجتمعية :**شارك أسئلتك وتجاربك في المنتديات التقنية العربية والدولية. المعرفة تنمو من خلال المشاركة.
- **المراجعة المنتظمة :**عد إلى الفصول السابقة كلما شعرت بالحاجة إلى ترسيخ المفاهيم أو اكتساب فهم أعمق.

نتمنى لك رحلة ممتعة ومثمرة في عالم بناء توزيعات لينكس، ونأمل أن تكون هذه الصفحات نقطة انطلاقك نحو إبداعات رقمية لا حدود لها.

الفصل الأول: أساسيات آرتش لينكس وبيئة العمل

1.1 نظرة تاريخية على لينكس وصعود آرتش

1.1.1نشأة نظام لينكس (1991) في أوائل التسعينيات، كان عالم أنظمة التشغيل تهيمن عليه أنظمة يونكس التجارية ونظام-MS DOSمن مايكروسوفت. كانت أنظمة يونكس قوية ولكنها باهظة الثمن ومغلقة المصدر، مما حد من إتاحتها للجامعات أو الشركات الكبيرة فقط.

في عام 1991، لم يكن لينوس تورفالدس، وهو طالب علوم حاسوب في جامعة هلسنكي بفنلندا، راضيًا عن نظام التشغيل مينيكس (وهو نسخة صغيرة من يونكس لأغراض تعليمية). قرر أن يكتب نواة (kernel) خاصة به كهواية وأطلقها عبر الإنترنت بترخيص مفتوح المصدر.

في إعلانه العام الأول، كتب تورفالدس: "أعمل على نظام تشغيل (مجاني) كهواية، لن يكون شيئًا كبيرًا أو احترافيًا مثل جنو (GNU)."

لكن سرعان ما انضم مئات المبرمجين من جميع أنحاء العالم لتطوير هذا المشروع الجديد، الذي سُمي "نواة لينكس Linux) " .(Kernelعندما تم دمجها مع أدوات مشروع جنو (الذي أسسه ريتشارد ستولمان)، حصلنا على ما نعرفه اليوم بنظام التشغيل جنو/لينكس.(GNU/Linux)

1.1.2بداية التوزيعات نظرًا لأن لينكس كان مجرد نواة بدون أدوات جاهزة للاستخدام، احتاج الناس إلى طريقة سهلة لتجميع النواة مع البرامج الأساسية وتوزيعها كنظام كامل. هكذا ولدت فكرة "التوزيعة.(Distribution) "

من بين أوائل التوزيعات كانت:

- سلاك وير :(1993) (Slackware) واحدة من أقدم التوزيعات، ولا تزال موجودة حتى اليوم.
- دبيان :(1993) (Debian) ركزت على الاستقرار والمجتمع المنظم .أصبحت فيما بعد الأساس للعديد من التوزيعات مثل أوبونتو.
- ريد هات لينكس :(1995) (Red Hat Linux) استهدفت الشركات، وأصبحت لاحقًا الأساس لـ RHEL و.Fedora

كان لكل توزيعة فلسفتها الخاصة: البعض ركز على سهولة الاستخدام (مثل أوبونتو لاحقًا)، والبعض على الاستقرار (مثل دبيان)، والبعض الآخر على التخصيص (مثل جنتو)

1.1.3ظهور آرتش لينكس (2002) في أوائل العقد الأول من القرن الحادي والعشرين، لاحظ مطور كندي يُدعى جود فينيت أن معظم التوزيعات الحالية إما معقدة للغاية (مثل جنتو، التي تتطلب بناء النظام من المصدر) أو "مُثقلة" لأنها تأتي مُحملة مسبقًا بإعدادات لا يحتاجها كل مستخدم.

في عام 2002، قرر جود فينيت إنشاء توزيعة جديدة أطلق عليها اسم "آرتش لينكس.(Arch Linux) "

- هدفه :توزيعة بسيطة، خفيفة الوزن، ومرنة.
- شعارها KISS :(- Keep It Simple, Stupid حافظ عليها بسيطة، يا غبي).
- لم تأت بواجهة رسومية؛ بل بدأت من سطر الأوامر.
- استخدمت مدير حزم جديدًا يسمى pacman، والذي كان سهلًا وعمليًا لإدارة الحزم.

لم يكن آرتش لينكس يستهدف المبتدئين، ولكنه جذب مجتمعًا من المستخدمين المتقدمين الذين أحبوا حرية التخصيص والشفافية الكاملة للنظام.

1.1.4المجتمع يتولى التطوير - آرون غريفين بعد بضع سنوات من قيادة المشروع، قرر جود فينيت التوقف عن تطوير آرتش لينكس لأسباب شخصية. في عام 2007، تولى آرون غريفين القيادة.

خلال فترة تولي غريفين:

- **أصبح تطوير آرتش أكثر تنظيمًا ومدفوعًا بالمجتمع.**
- **ازدهرت "ويكي آرتش (Arch Wiki) "وأصبحت واحدة من أهم مصادر المعرفة في عالم لينكس.**
- **ظهرت مشاريع مشتقة من آرتش، مثل ArchBang و.Manjaro**

اليوم، يُدار آرتش لينكس بواسطة فريق من المطورين الأساسيين ومجتمع ضخم من المساهمين، مما جعله واحدًا من أقوى توزيعات لينكس وأكثرها تأثيرًا.

ملخص هذا القسم:

- **بدأ لينكس كهواية لطالب جامعي في عام 1991.**
- **سرعان ما أصبح نظامًا عالميًا بفضل فلسفة المصدر المفتوح.**
- **ظهرت التوزيعات لتسهيل استخدامه، ولكل منها فلسفتها الخاصة.**
- **جاء آرتش لينكس (2002) كحل وسط: ليس معقدًا مثل جنتو، وليس مقيدًا مثل أوبونتو.**
- **اليوم، آرتش ليس مجرد توزيعة، بل هو مدرسة تعلم المستخدم كيفية عمل النظام من الداخل والخارج.**

ما الذي يجعل آرتش لينكس فريدًا؟

عندما نتحدث عن توزيعات لينكس، غالبًا ما تتبادر إلى الذهن مصطلحات مثل "سهولة الاستخدام"، "الاستقرار"، أو "الدعم الفني". ولكن مع آرتش لينكس، الوضع مختلف؛ إنه ليس مجرد نظام تشغيل آخر، بل هو فلسفة كاملة للتفاعل مع أجهزة الكمبيوتر. لفهم ما يميز آرتش، يجب أن ننظر إلى عدة مجالات رئيسية:

1.2.1 البساطة (Simplicity)

الشعار الأساسي لآرتش هو KISS (Keep It Simple, Stupid). ومع ذلك، فإن كلمة "البساطة" هنا لا تعني أن النظام سهل للمبتدئين. بل تعني:

- **تجنب تعقيد التصميم غير الضروري.**
- **عدم إضافة أدوات أو واجهات غير ضرورية.**
- **توفير الحد الأدنى فقط وترك الباقي للمستخدم.**

على سبيل المثال، تأتي توزيعات مثل أوبونتو مثبتة مسبقًا بواجهة جنوم (GNOME) ومتصفح فايرفوكس (Firefox) ومجموعة ليبر أوفيس (LibreOffice). في المقابل، عندما تقوم بتثبيت آرتش لأول مرة، فإنه يمنحك نظامًا أساسيًا فارغًا تقريبًا. أنت تقرر: ما هي الواجهة الرسومية التي تريدها؟ أي متصفح؟ أي أدوات مكتبية؟ هذا يجعل آرتش أشبه بقطع الليجو: أنت تبني النظام قطعة قطعة وفقًا لرغباتك.

1.2.2 الشفافية (Transparency)

آرتش لا يخفي شيئًا عن المستخدم:

- **جميع ملفات التكوين موجودة كنصوص قابلة للقراءة والتحرير.**
- **لا يوجد "سحر" خلف الكواليس.**
- **حتى عملية التثبيت نفسها هي سلسلة من الأوامر التي يكتبها المستخدم، مما يسمح له بفهم كيفية بناء نظام التشغيل خطوة بخطوة.**

على سبيل المثال، في أوبونتو، يمكن للمستخدم إضافة مستودع برامج عبر واجهة رسومية أو أمر بسيط مثل add-apt-repository. في آرتش، يتم ذلك يدويًا عن طريق تحرير الملف /etc/pacman.conf، مما يجعلك ترى وتفهم بالضبط ما يحدث.

1.2.3المرونة(Flexibility)

آرتش لا يفرض عليك أي قرارات:

- **هل تريد استخدامKDE Plasma ؟ يمكنك تثبيته.**
- **هل تفضل GNOME أوXFCE ؟ كلاهما ممكن.**
- **لا تريد واجهة رسومية على الإطلاق وتفضل مدير نوافذ بسيط مثل i3wm أوbspwm ؟ كل ذلك متاح بالكامل.**

هذه المرونة تجعل آرتش مناسبًا لمختلف المستخدمين، من عشاق السرعة والبصمة الصغيرة إلى المتحمسين للأشكال والواجهات المتقدمة.

1.2.4الإصدار المستمر(Rolling Release)

واحدة من أكبر الاختلافات بين آرتش والتوزيعات الأخرى هو نظام التحديث:

- **معظم التوزيعات (مثل أوبونتو أو فيدورا) تصدر إصدارات جديدة كل 6 أشهر إلى سنة. يحتاج المستخدم إلى الترقية بين الإصدارات.**
- **آرتش، ومع ذلك، هو "إصدار مستمر(Rolling Release) "، مما يعني أن التوزيعة محدثة دائمًا .التثبيت مرة واحدة لآرتش يكفي لسنوات؛ تحتاج فقط إلى إبقائها محدثة عبر .sudo pacman -Syu**

المزايا:

- **تحصل دائمًا على أحدث إصدارات البرامج والنواة.**
- **لا تحتاج إلى إعادة تثبيت النظام.**

العيوب:

- **قد يتسبب تحديث غير متوافق في بعض الأحيان في حدوث مشاكل مفاجئة.**
- **يحتاج المستخدم إلى الانتباه إلى أخبار آرتش (Arch News) قبل التحديث.**

1.2.5مدير الحزمPacman

أحد ركائز آرتش لينكس هو مدير الحزم .pacman

- **مصمم ليكون بسيطًا وسريعًا.**
- **يتعامل تلقائيًا مع التبعيات.(dependencies)**
- **الأوامر موحدة ويسهل تذكرها.**

أمثلة شائعة:

Bash

#تحديث النظام

sudo pacman -Syu

#تثبيت برنامج

sudo pacman -S firefox

#إزالة برنامج مع ملفاته الإضافية

sudo pacman -Rns firefox

#البحث عن برنامج

pacman -Ss vlc

مقارنة:

- **aptفي دبيان/أوبونتو قد يتطلب أوامر أكثر تعقيدًا.**
- **dnfفي فيدورا أبطأ أحيانًا في الأداء.**

1.2.6 Arch Wiki:موسوعة المعرفة

واحدة من أهم الأشياء التي تميز آرتش ليست التوزيعة نفسها، بل وثائقها.

- **تعتبر "ويكي آرتش (Arch Wiki) "واحدة من أكبر وأشمل مراجع لينكس على الإطلاق.**
- **حتى مستخدمو التوزيعات الأخرى (دبيان، فيدورا، مانجارو) يعتمدون عليها لحل المشكلات.**
- **تشرح كل شيء، من تثبيت بطاقة رسوميات إلى إعداد خادم ويب كامل.**

1.2.7 AUR (Arch User Repository)

لدى آرتش مستودعات رسمية ضخمة، لكن قوته الحقيقية تظهر مع:AUR

- **هو مستودع تم بناؤه بالكامل بواسطة المجتمع.**
- **يحتوي على آلاف الحزم التي لن تجدها في المستودعات الرسمية.**

- **يتم إدارته عبر ملفات PKGBUILDالتي تسمح لك ببناء الحزمة على جهازك.**

على سبيل المثال، يمكن العثور بسهولة على برنامج غير متاح رسميًا، مثل جوجل كروم(Google Chrome) ، في AUR عبر أداة مثل :yay

Bash

yay -S google-chrome

ولكن يجب أن تكون حذرًا:

- **ليست كل الحزم في AUR مضمونة الجودة العالية أو الأمان.**
- **يوصى دائمًا بقراءة ملف PKGBUILDقبل التثبيت.**

1.2.8آرتش كأساس لتوزيعات أخرى

قوة آرتش جعلته أساسًا لعدد من التوزيعات المشتقة:

- **مانجارو :(Manjaro) تقدم آرتش مع واجهة رسومية جاهزة للاستخدام للمبتدئين.**
- **إنديفوروس أو إس :(EndeavourOS) توفر تجربة أقرب لآرتش ولكن مع تثبيت أسهل.**
- **جارودا لينكس :(Garuda Linux) تركز على الرسوميات والأداء.**
- **حلوان لينكس) (Helwan Linux) مثالنا :(توزيعة مصرية مبنية على آرتش بنكهة محلية وهوية فريدة.**

هذا يثبت أن آرتش ليس مجرد نظام تشغيل، بل هو منصة لبناء أنظمة أخرى.

ملخص القسم 1.2:

ما يميز آرتش ليس فقط التكنولوجيا التي بُني عليها، بل الفلسفة التي يتبناها:

- **البساطة،**
- **الشفافية،**
- **المرونة،**
- **الإصدار المستمر(Rolling Release) ،**
- **قوة المجتمع،**
- **والاعتماد على المستخدم كعنصر أساسي في بناء تجربته.**

1.2 الفلسفة العميقة لآرتش لينكس

1.3

آرتش لينكس ليس مجرد توزيعة خفيفة أو مرنة؛ بل هو مدرسة كاملة في فلسفة تصميم أنظمة التشغيل. لفهم "روح" آرتش، يجب أن نتعمق في المبادئ التي توجهه.

1.3.1 مبدأ) KISS ابقها بسيطة، أيها الغبي(

كلمة "بساطة" هنا لا تعني سهولة الاستخدام. آرتش لا يحاول أن يكون سهلًا للمبتدئين مثل أوبونتو أو منت. بل يعني الوضوح وغياب التعقيد: لا توجد طبقات خفية من البرامج الوسيطة (middleware) أو أدوات إدارة تلقائية تفرض نفسها عليك.

- على سبيل المثال:
  - في أوبونتو، عندما تقوم بتثبيت تعريف لبطاقة الرسوميات، توجد أدوات رسومية مخصصة تقوم بالمهمة نيابة عنك.
  - في آرتش، يتم ذلك عن طريق تثبيت الحزم المناسبة يدويًا من pacmanأو مستودع AUR وتعديل ملفات الإعدادات المحددة.

النتيجة: النظام أبسط من الداخل، لكنه يتطلب المزيد من المعرفة والخبرة من المستخدم.

1.3.2 التحكم الكامل (محورية المستخدم(

يُبنى آرتش لينكس على فكرة أن المستخدم هو الأدرى باحتياجاته.

- النظام لا يفرض عليك حزمًا أو إعدادات معينة.
- حتى عملية التثبيت لا توفر واجهة رسومية، بل تمنحك الأدوات الأساسية لبناء النظام بنفسك.

هذا التحكم الكامل يجعل آرتش مثاليًا للمطورين والمهندسين الذين يحتاجون إلى بيئة عمل مخصصة. فمثلاً، في أوبونتو، يتم تثبيت النظام مع واجهة GNOME بشكل افتراضي. أما في آرتش، فبعد التثبيت الأولي، لا توجد لديك حتى واجهة رسومية؛ بل شاشة سوداء (TTY).إذا أردتGNOME ، تقوم بتثبيتها بنفسك؛ وإذا أردت KDE أو XFCE أو حتى لا شيء، فالقرار لك وحدك.

1.3.3 الشفافية

أحد المبادئ الأساسية في آرتش هو أن كل شيء يجب أن يكون واضحًا ومفهومًا.

- لا يوجد "سحر" يحدث في الخلفية.
- جميع الإعدادات يمكن تعديلها عبر ملفات نصية بسيطة.
- التوثيق (Arch Wiki) يشرح كل خطوة بالتفصيل.

على سبيل المثال، إذا أردت تشغيل خدمة في آرتش، فإنك تستخدم systemctl enable...وتفهم ما يحدث في الخلفية. في حين أنك في توزيعات أخرى قد تضغط على زر في واجهة رسومية ولا تعرف ما الذي تم خلف الكواليس.

1.3.4التعلم بالممارسة

يختلف آرتش لينكس عن معظم التوزيعات لأنه يعلّمك أثناء استخدامه:

- **عملية التثبيت نفسها هي درس عملي في كيفية عمل نظام لينكس.**
- **إعدادات الشبكة، المستخدمين، ومدير الإقلاع (boot manager) هي كلها خطوات يمر بها المستخدم ويفهمها.**
- **بمرور الوقت، يتحول مستخدم آرتش من مجرد "مستهلك" للنظام إلى "متحكم" فيه.**

يصف بعض المستخدمين التجربة بأنها: "آرتش لا يعطيك سمكة، بل يعلّمك كيف تصطاد".

1.3.5نموذج التحديث المستمر كجزء من الفلسفة

لم يكن اختيار نظام التحديث المستمر (Rolling Release) لآرتش قرارًا عشوائيًا، بل نابعًا من فلسفته:

- **لماذا يجبر المستخدم على إعادة تثبيت النظام كل ئة أشهر أو سنة، كما في أوبونتو؟**
- **بدلاً من ذلك، دع المستخدم يحصل على أحدث البرامج فورًا، ويستمر نظامه في التطور معه.**

هذا يعكس إيمان آرتش بفكرة أن النظام يجب أن يكون حيًا دائمًا، لا يشيخ أو يصبح قديمًا.

1.3.6التطور المجتمعي

لا يسعى آرتش لإرضاء الشركات أو التوجه التجاري. بل على العكس:

- **المجتمع هو القلب النابض للتوزيعة.**
- **معظم الحلول تجدها فيArch Wiki ، الذي كتبه المستخدمون.**
- **مستودع AUR مبني بالكامل على مساهمات المجتمع.**

يعكس هذا مبدأً فلسفيًا هامًا: المعرفة جماعية وليست حكرًا على مؤسسة أو شركة.

1.3.7آرتش ليس للجميع (فلسفة انتقائية(

لم يهدف مؤسس آرتش إلى أن تكون التوزيعة سهلة أو مناسبة للجميع.

- **إذا كنت مبتدئًا تمامًا، قد تجد آرتش صعبًا جدًا.**
- **لكن إذا أردت فهم لينكس من الداخل والتحكم به، ستجد آرتش هو الخيار الأفضل.**

هذا المبدأ جعل آرتش يُعرف أحيانًا بأنه توزيعة "نخبوية"، ليس بمعنى المتعالي، بل بمعنى أنه يتطلب مستوى معينًا من الجدية وحب الاستطلاع.

خلاصة القسم 1.3 :يمكن تلخيص فلسفة آرتش لينكس في:

- **بساطة التصميم دون تعقيد غير ضروري.**
- **التحكم الكامل بالنظام.**
- **الشفافية في كل شيء.**
- **التعلم بالممارسة.**
- **التحديث المستمر الذي يواكب العصر.**
- **مجتمع قوي يدفع عجلة التطوير والدعم.**

هذه الفلسفة جعلت من آرتش أكثر من مجرد توزيعة: إنه طريقة تفكير وتعامل مع أنظمة التشغيل.

1.4نموذج التحديث المستمر(Rolling Release)

إحدى أبرز ميزات آرتش لينكس، وما يجعله مختلفًا عن غالبية التوزيعات الأخرى، هي نظام التحديث المستمر، المعروف باسم Rolling Release.هذا المفهوم ليس مجرد طريقة لتوزيع البرامج؛ بل هو جزء أساسي من فلسفة آرتش.

1.4.1ما هو التحديث المستمر(Rolling Release) ؟

في عالم البرامج، توجد طريقتان رئيسيتان لتوزيع الإصدارات:

- الإصدار الثابت:(Fixed Release)
  - تقوم التوزيعة بإصدار نسخة جديدة على فترات زمنية ثابتة (مثل أوبونتو كل 6 أشهر، أو دبيان كل سنتين.(
  - البرامج داخل هذا الإصدار تبقى بشكل عام كما هي، باستثناء التحديثات الأمنية.
  - مثال: أوبونتو 22.04 سيستمر في استخدام نفس إصدار GNOME والنواة حتى إصدار 24.04.
- الإصدار المستمر:(Rolling Release)
  - النظام لا يمتلك "إصدارات رئيسية"، بل يتم تحديث الحزم بشكل مستمر.
  - عندما يتم إصدار نسخة جديدة من النواة أو أي برنامج، يتم تحديثها فورًا في المستودعات.
  - مثال: آرتش لينكس يمتلك دائمًا أحدث إصدار من نواة لينكس، دون انتظار "إصدار جديد من آرتش."

1.4.2مميزات التحديث المستمر

- دائمًا أحدث البرامج :تحصل على أحدث إصدار من محرر النصوص المفضل لديك أو بيئة البرمجة بمجرد صدورها. هذا مناسب جدًا للمطورين الذين يحتاجون إلى بيئة حديثة باستمرار.
- لا حاجة لإعادة التثبيت :مع التوزيعات ذات الإصدارات الثابتة، قد تضطر إلى إعادة تثبيت النظام أو ترقيته من حين لآخر. أما مع آرتش، فعملية تثبيت واحدة تكفي لسنوات؛ كل ما عليك هو الاستمرار في التحديث.
- نظام دائم الحيوية :آرتش لا يشيخ أبدًا. طالما أنك تقوم بتحديثه، فإنه سيظل دائمًا في أحدث صورة له.

1.4.3عيوب التحديث المستمر

- احتمالية حدوث أعطال :قد يتسبب تحديث غير متوافق أو وجود خطأ في إحدى الحزم في مشكلة. على سبيل المثال، قد يتسبب تحديث لتعريف بطاقة الرسوميات أحيانًا في مشاكل في الإقلاع.
- مسؤولية أكبر على المستخدم :يجب عليك متابعة أخبار آرتش باستمرار للتأكد من وجود أي تنبيهات مهمة قبل التحديث، وأن تكون مستعدًا لإصلاح المشكلات بنفسك.

- **استهلاك أكبر للإنترنت :أنت تقوم دائمًا بتنزيل إصدارات جديدة من البرامج.**

1.4.4كيفية تعامل آرتش مع التحديث المستمر

على الرغم من العيوب، يمتلك آرتش نظامًا قويًا لتقليل المشاكل:

- **فحص الحزم :قبل أن تدخل أي حزمة إلى المستودع الرسمي، يتم فحصها في مستودع مخصص للتجربة .[testing]**
- **التوثيق :يتم توثيق أي مشكلة كبيرة فورًا على Arch Wiki أو على صفحة الأخبار.**
- **دعم المجتمع :يشارك المستخدمون الحلول بسرعة على المنتديات أو موقع.Reddit**

1.4.5استراتيجيات لتحديث آمن

- **التحديث بانتظام :عدم التحديث لفترات طويلة يمكن أن يجعل ترقية النظام صعبة بسبب تعارضات كثيرة. من الأفضل التحديث أسبوعيًا أو كل أسبوعين.**
- **استخدام Timeshift أو النسخ الاحتياطية :يمكنك أخذ نسخة احتياطية قبل التحديث للعودة إليها في حالة حدوث مشكلة.**
- **قراءة الأخبار قبل التحديث :يبدو أمر sudo pacman -Syuبسيطًا، لكن يجب أن تكون على دراية بما قد يتغير بعده.**

1.4.6مقارنة مع توزيعات أخرى

فيدورا (شبه مستمر)	أوبونتو/دبيان (ثابت)	آرتش (مستمر)	المعيار
تحديثات سريعة نسبيًا	مستقر طوال فترة الإصدار	أحدث إصدار دائمًا	تحديثات النواة
متوسط	أعلى (مناسب للخوادم)	أقل (لكنه مرن)	استقرار النظام
6~أشهر	6أشهر أو أكثر	لا يوجد (مستمر)	الفاصل بين الإصدارات
متوسط	أسهل للمبتدئين	يحتاج إلى متابعة مستمرة	سهولة الإدارة

1.4.7أمثلة عملية

- **مطور ألعاب :يحتاج إلى أحدث مكتبات Vulkan و ← Mesa آرتش هو الأنسب.**
- **شركة خوادم :تحتاج إلى استقرار طويل الأمد بدون مفاجآت ← دبيان أو RHEL أفضل.**
- **مستخدم عادي :يريد جهازه يعمل دائمًا دون قلق ← مانجارو (مبني على آرتش ولكنه أكثر استقرارًا.(**

خلاصة القسم 1.4 :نموذج التحديث المستمر في آرتش لينكس هو ميزة قوية تجعله دائمًا عصريًا ومواكبًا. لكنه سلاح ذو حدين: يمنحك الحرية والحداثة، ولكنه يتطلب مسؤولية ويقظة.

1.5مدير الحزمPacman

أحد الأعمدة الأساسية في آرتش لينكس، وما يميزه عن التوزيعات الأخرى، هو مدير الحزم .Pacman في أنظمة لينكس، مدير الحزم هو الأداة التي تسمح لك بتنزيل البرامج وتثبيتها وتحديثها وإزالتها من مستودعات التوزيعة. لكن Pacman يبرز بفضل فلسفته البسيطة وأدائه القوي.

1.5.1ما هوPacman ؟

- اسم "Pacman" هو اختصار لـ) Package Managerمدير الحزم.(
- هو الأداة الرسمية لإدارة الحزم في آرتش لينكس.
- مكتوب بلغة C ليكون سريعًا وخفيفًا.
- يدير الحزم التي تأتي بامتداد) pkg.tar.zstملفات مضغوطة تحتوي على البرامج.(

1.5.2مميزاتPacman

- أوامر بسيطة:
  - الأوامر قصيرة وسهلة التذكر.
  - أمثلة S- :للتثبيت، R-للإزالة، Q-للاستعلام.
- إدارة تلقائية للتبعيات:
  - إذا احتاج برنامج لمكتبات إضافية، يقوم Pacman بتثبيتها تلقائيًا.
- السرعة:
  - بفضل تصميمه بلغة C ونظام مستودعات يعتمد على الملفات المضغوطة.
- التوحيد:
  - تُستخدم نفس الأداة لكل شيء (التثبيت، التحديث، البحث، الإزالة.(

1.5.3أوامر Pacman الأساسية

الأمر	الوظيفة	مثال
pacman -S package	تثبيت برنامج	pacman -S firefox
pacman -R package	إزالة برنامج	pacman -R vlc
pacman -Rns package	إزالة برنامج + التبعيات غير المستخدمة	pacman -Rns gimp
pacman -Ss keyword	البحث عن برنامج في المستودعات	pacman -Ss vlc
pacman -Qs keyword	البحث عن برنامج مثبت محليًا	pacman -Qs python
pacman -Qi package	عرض معلومات عن برنامج مثبت	pacman -Qi nano
pacman -Syu	تحديث النظام بالكامل	pacman -Syu

1.5.4ملفات إعدادPacman

- **الملف الرئيسي/etc/pacman.conf :**
  - **يحتوي على إعدادات مثل المستودعات المفعلة، خيارات التثبيت، وإدارة التوقيعات الرقمية.**
- **قاعدة البيانات المحلية/var/lib/pacman/ :**
  - **حيث يخزن Pacman معلومات عن الحزم المثبتة.**

مثال من :pacman.conf

```
[options]
HoldPkg = pacman glibc
Architecture = auto
CheckSpace
SigLevel = Required DatabaseOptional

[core]
Include = /etc/pacman.d/mirrorlist

[extra]
Include = /etc/pacman.d/mirrorlist

[community]
Include = /etc/pacman.d/mirrorlist
```

1.5.5التوقيعات الرقمية(Package Signing)

- **يستخدم Pacman نظام GPG للتحقق من صحة الحزم.**
- **هذا يعني أن كل حزمة لها توقيع رقمي لضمان عدم تغييرها أثناء النقل.**
- **في حالة حدوث خطأ في المصادقة، لن يسمح Pacman بالتثبيت إلا إذا أجبرت النظام (وهو أمر غير مستحسن.(**

1.5.6مقارنة Pacman بمديري حزم آخرين

الميزة	Pacman (Arch)	APT (Debian)	DNF (Fedora)
لغة البرمجة	C (سريع جدًا)	C++	Python/C
صعوبة الأوامر	سهلة وبسيطة	متوسط	متوسط
التبعيات	إدارة قوية	قوية	قوية
التحديثات	دائمًا مستمر(Rolling)	ثابت حسب الإصدارات	نصف سنوية
السرعة	أداء أسرع	أبطأ نسبيًا	أبطأ منPacman

1.5.7استخدام Pacman مع مستودعات إضافية

يمكنك إضافة مستودعات إضافية عن طريق تعديل ملف .etc/pacman.conf/

مثال: إضافة مستودع) multilib لتشغيل برامج 32 بت:(

[multilib]

Include = /etc/pacman.d/mirrorlist

ثم قم بتشغيلSyu- sudo pacman :

1.5.8مشاكل شائعة مع Pacman وحلولها

- قاعدة بيانات تالفة:
  - استخدم) sudo pacman -Syyلإجبار تحديث قاعدة البيانات.(
- ملفات متعارضة أثناء التحديث:
  - الحل: قم بإزالة الملف يدويًا أو استخدم خيار .overwrite--
- انقطاع الإنترنت أثناء التحديث:
  - يمكن استئناف العملية بسهولة بعد إعادة الاتصال.

1.5.9العلاقة بين Pacman وAUR

بينما Pacman قوي جدًا، إلا أنه لا يدير AUR (Arch User Repository) مباشرة.

- **بالنسبة لحزمAUR ، تستخدم أدوات مساعدة مثل yayأو paru.**
- **هذه الأدوات تعتمد في النهاية على Pacman ولكنها تضيف خطوة إنشاء الحزمة من ملف.PKGBUILD**

خلاصة القسم Pacman 1.5 :ليس مجرد مدير حزم عادي؛ إنه قلب تجربة آرتش لينكس .بساطته وسرعته ومرونته تجعله واحدًا من أسرع وأقوى مديري الحزم في عالم لينكس.

1.6مستودع مستخدمي آرتش(AUR)

تُعدّ قوة آرتش لينكس الكبيرة في مستودعAUR ، والذي يعتبر أحد أضخم المستودعات المجتمعية في عالم لينكس.

1.6.1ما هوAUR ؟

- AURهو اختصار لـ (Arch User Repositoryمستودع مستخدمي آرتش.(
- إنه مستودع ضخم يحتوي على "وصفات" الحزم (PKGBUILDs) مكتوبة بواسطة المجتمع.
- هدفه هو تمكين المستخدمين من تثبيت البرامج غير المتوفرة في المستودعات الرسمية بسهولة.

المستودعات الرسمية تحتوي فقط على البرامج التي يختبرها فريق آرتش رسميًا. أماAUR ، فهو مساحة مفتوحة حيث يمكن للمستخدمين مشاركة أي برنامج أو أداة أو حتى سمة.

1.6.2ما هو ملفPKGBUILD ؟

- ملف PKGBUILD هو برنامج نصي نصي مكتوب بلغة.Bash
- يحتوي على تعليمات لبناء حزمة من المصدر أو من ملفات مجمعة مسبقًا.
- Pacmanلا يتعامل مع AUR مباشرة؛ يقوم المستخدم ببناء الحزم بنفسه باستخدام هذا الملف.

مثال بسيط:

```
Bash

pkgname=hello

pkgver=1.0

pkgrel=1

arch=('x86_64')

source=("http://example.com/$pkgname-$pkgver.tar.gz")

md5sums=('SKIP')

build()

{

 cd "$srcdir/$pkgname-$pkgver"

 ./configure --prefix=/usr
```

```
 make
}
package()
{
 cd "$srcdir/$pkgname-$pkgver"
 make DESTDIR="$pkgdir/" install
}
```

1.6.3 لماذا يعتبر AUR مهمًا؟

- **تغطية واسعة :أي برنامج يخطر ببالك غالبًا ما تجده في.AUR**
- **مجتمع ضخم :آلاف المساهمين يرفعون ويحدثون الحزم يوميًا.**
- **مرونة :يمكنك تعديل ملف PKGBUILD بنفسك قبل البناء.**
- **سرعة التوفر :غالبًا ما تُرفع البرامج إلى AUR قبل وصولها إلى المستودعات الرسمية (إن وصلت أبدًا.)**

1.6.4 كيف يعمل AUR عمليًا؟

1. **تبحث عن الحزمة على موقعAUR: https://aur.archlinux.org**
2. **تنسخ ملف.PKGBUILD**
3. **تستخدم الأمر makepkg -si لبناء الحزمة وتثبيتها محليًا.**

1.6.5أدوات مساعدة لـAUR

نظرًا لأن التعامل اليدوي مع PKGBUILDs ممل، فقد ابتكر المجتمع أدوات لتبسيط هذه العملية. أشهرها:

الأداة	الميزة
yay	أشهر مساعد؛ يعمل مع Pacman و.AUR
paru	مشابه لـ yay ولكنه بواجهة أبسط.
trizen	يدعم ميزات البحث والبناء المتقدمة.
pamac	واجهة مستخدم رسومية (GUI) تشبه مدير الحزم في مانجارو.

مثال باستخدام yay: yay -S google-chrome هذا الأمر سيبحث فيAUR ، ويبني الحزمة، ويقوم بتثبيتها تلقائيًا.

1.6.6تحديات ومخاطرAUR

- الأمان :بما أن الحزم مكتوبة بواسطة المجتمع، فقد تحتوي على شيفرة خبيثة.
  - الحل :قم دائمًا بفحص ملف PKGBUILD قبل البناء.
- جودة الحزم :ليست كل الحزم في AUR محدثة أو مستقرة.
- الاعتماديات :(Dependencies) أحيانًا، توجد اعتماديات غير متوفرة في المستودعات الرسمية.

1.6.7العلاقة بين AUR والمستودعات الرسمية

- يمكن لحزمة من AUR أن تُعتمد لاحقًا وتُنقل إلى المستودعات الرسمية.
- مثال :العديد من البرامج بدأت حياتها في AUR وأصبحت رسمية بعد اكتسابها شعبية.

1.6.8مقارنة AUR مع مستودعات مشابهة في توزيعات أخرى

التوزيعة	النظام المعادل	مقارنة بـAUR
Debian	PPA (Personal Package Archives)	مشابه، لكنه أصغر بكثير حجمًا ومحتوى.
Fedora	COPR	مشابه جدًا، لكنه أقل انتشارًا.
openSUSE	OBS (Open Build Service)	نظام شامل ولكنه أكثر تعقيدًا من.AUR

1.6.9أمثلة لحزم AUR الشهيرة

- google-chrome:غير متوفر رسميًا بسبب الترخيص.
- spotify:مغلق المصدر، ولكنه متاح عبر.AUR
- visual-studio-code-bin:النسخة الرسمية المجمعة مسبقًا من.Microsoft VSCode
- whatsapp-nativefier:لتحويل WhatsApp إلى تطبيق سطح مكتب.

خلاصة القسم 1.6 :يُعدّ AUR الركيزة الثانية بعد Pacman التي تجعل من آرتش لينكس نظامًا فريدًا. بفضل هذا المستودع، يحصل المستخدم على إمكانية الوصول إلى آلاف البرامج الإضافية غير الرسمية، مما يفتح الباب لحرية ومرونة هائلة، ولكنه يتطلب مسؤولية وفحصًا دقيقًا من المستخدم.

1.7فلسفة آرتش لينكس: مبدأKISS (Keep It Simple, Stupid)

أحد أهم ركائز آرتش لينكس هو فلسفةKISS ، التي تقف اختصارًا لـ) "Keep It Simple, Stupid" :ابقها بسيطة، أيها الغبي.(

على الرغم من أن العبارة قد تبدو ساخرة في البداية، إلا أنها تحمل رؤية عميقة لبناء أنظمة البرمجيات :البساطة أقوى من التعقيد.

1.7.1ما الذي تعنيه البساطة في آرتش؟

البساطة لا تعني نقص الميزات. بل تعني أن النظام مبني على مكونات صغيرة، واضحة، وقابلة للفهم. آرتش لا يحاول إخفاء التعقيد عن المستخدم (كما تفعل توزيعات مثل أوبونتو أو فيدورا)؛ بل يضع الأدوات أمامك ويمنحك التحكم الكامل.

1.7.2تجليات مبدأ KISS في آرتش لينكس

- ملفات الإعدادات النصية :لا توجد أدوات رسومية معقدة لتغيير الإعدادات. كل شيء تقريبًا يتم عبر ملفات نصية قابلة للقراءة والفهم (مثل systemd، و .(pacman.conf
- إدارة الحزم :(Pacman) أداة واحدة قوية تتولى التثبيت والتحديث والإزالة. لا توجد عشرات الأدوات المختلفة كما في بعض التوزيعات الأخرى.
- التركيز على النواة :يمنحك آرتش نظامًا أساسيًا نظيفًا. والباقي متروك لك لبنائه خطوة بخطوة.
- التوثيق :(Arch Wiki) بدلاً من تطوير أدوات رسومية تخفي التفاصيل، يركز فريق آرتش على توثيق كل شيء بوضوح.

1.7.3لماذا يعتبر KISS مهمًا للمستخدم؟

- التحكم الكامل :أنت تعرف بالضبط ما هو موجود في نظامك.
- سهولة الصيانة :أي مشكلة يمكن تتبعها بسهولة لأن كل شيء واضح ومباشر.
- المرونة :يصبح النظام مثل صندوق الأدوات الذي تبني منه فقط ما تحتاجه.
- التعلم :مبدأ KISS يجعل آرتش منصة تعليمية ممتازة لفهم لينكس.

1.7.4الفرق بين KISS والتبسيط الزائف

بعض التوزيعات تحاول أن تكون "سهلة" عبر بناء طبقات رسومية تخفي التعقيد. هذا يؤدي إلى "تبسيط زائف"، حيث يفقد المستخدم القدرة على التحكم الكامل بالنظام، وعندما يحدث خطأ، يصبح من الصعب إصلاحه.

آرتش، على الجانب الآخر، يتبع :KISS يبقي النظام بسيطًا ولكنه شفاف.

1.7.5أمثلة عملية لمبدأ KISS في آرتش

- تثبيت آرتش :لا يوجد "مثبت رسومي" معقد. أنت تختار الأقراص، والتقسيم، والنواة، وبيئة سطح المكتب بنفسك.
- إعداد الشبكة :بدلاً من أداة رسومية ضخمة، يمكنك الاعتماد على أدوات بسيطة مثل ipأو .systemd-networkd
- بناء الحزم :العملية واضحة وبسيطة من خلال ملف .PKGBUILD

1.7.6انتقادات لمفهوم KISS في آرتش

على الرغم من أن KISS هو نقطة قوة، إلا أن البعض ينتقده:

- منحنى تعليمي حاد :قد يجد المبتدئ الأمر صعبًا جدًا في البداية.
- عمل متكرر :أحيانًا تحتاج إلى إعداد أشياء يدوية كان من الممكن أن تكون مؤتمتة.
- وقت مستهلك :تخصيص آرتش يستغرق وقتًا أطول من التوزيعات "الجاهزة" مثل أوبونتو.

ومع ذلك، هذه الانتقادات هي في الواقع جزء من فلسفة آرتش: إذا كنت تريد الراحة المطلقة، فربما آرتش ليس لك.

1.7.7 KISSعلى لسان مؤسس آرتش

قال مؤسس آرتش، جود فينت، في مقابلة قديمة: "آرتش ليس للجميع. إنه للمستخدم الذي يريد أن يتعلم ويتحكم، وليس لمن يريد كل شيء جاهزًا".

هذا يلخص الفكرة :آرتش = حرية + بساطة + مسؤولية.

خلاصة القسم 1.7 :فلسفة KISS هي ما يجعل آرتش لينكس فريدًا بين التوزيعات. إنها فلسفة بساطة واضحة تمنح المستخدم المرونة والتحكم الكامل على حساب منحنى تعليمي أكثر حدة. آرتش لا يعد بالسهولة المطلقة، ولكنه يعد بالوضوح والشفافية.

1.8مجتمع آرتش وArch Wiki

لا يمكنك الحديث عن آرتش لينكس دون ذكر المجتمع الذي يقف خلفه. على الرغم من أن آرتش بدأ كتوزيعة صغيرة أسسها جود فينت في عام 2002، فإن سر قوته المستمرة اليوم هو المجتمع النشط الذي يقوم بصيانتها وتطويرها.

1.8.1قوة مجتمع آرتش

- **آرتش لينكس ليس مشروعًا تابعًا لشركة؛ بل هو مشروع مجتمعي بالكامل.**
- **يساهم آلاف المطورين والمستخدمين يوميًا من خلال:**
  - **إصلاح الأخطاء.**
  - **تحديث الحزم.**
  - **إضافة التوثيق.**
  - **دعم المستخدمين الجدد.**
- **يعمل المجتمع بشفافية تامة: جميع المناقشات مفتوحة، وجميع القرارات متاحة على القوائم البريدية والمنتديات.**

1.8.2 Arch Wiki:موسوعة لينكس الأولى

- **Arch Wikiهو التوثيق الرسمي للمشروع.**
- **بدأ كصفحات بسيطة تشرح كيفية تثبيت آرتش، ولكنه اليوم يعتبر أكبر وأشمل مصدر توثيق لأنظمة لينكس بشكل عام، حتى أن مستخدمي التوزيعات الأخرى يعتمدون عليه.**

ما الذي يجعل Arch Wiki مميزًا؟

- **التفصيل :كل خطوة مشروحة بوضوح.**
- **التحديث المستمر :أي تغيير في الحزم أو النظام يتم عكسه بسرعة في الويكي.**
- **الشمولية :لا يقتصر على آرتش؛ بل يضم معلومات عن النظام ككل (النواة، systemd، الشبكات، Xorg، Wayland، إلخ.(**
- **مجاني ومفتوح :مفتوح للجميع، ويمكن لأي مستخدم أن يساهم فيه.**

1.8.3 المجتمع كمصدر للدعم

- **المنتديات :(Arch Forums) مكان لتبادل الأسئلة والمشاكل والحلول.**
- **قنوات IRC و :Matrix للتواصل المباشر مع المطورين والمستخدمين المخضرمين.**
- **:AUR (Arch User Repository)مستودع ضخم أنشأه المجتمع، يحتوي على مئات الآلاف من الحزم غير الموجودة في المستودعات الرسمية.**

1.8.4 روح المشاركة والتعلم

- **في آرتش، الفلسفة ليست فقطKISS ، بل أيضًا "علّم غيرك."**
- **المستخدم الجديد الذي يستفيد من المنتدى أو الويكي غالبًا ما يعود لاحقًا ليساعد الآخرين.**
- **هذه الدورة المستمرة من التعلم والمشاركة جعلت آرتش أكثر من مجرد توزيعة: إنها مدرسة في لينكس.**

1.8.5 انتقادات للمجتمع

- **يُوصف المجتمع أحيانًا بأنه صارم مع المبتدئين.**
- **بعض الردود قد تكون قاسية إذا لم يقرأ المستخدم التوثيق أولًا.**
- **لكن السبب هو أن فلسفة آرتش مبنية على الاعتماد على الذات والقراءة قبل طلب المساعدة.**

1.8.6 دروس مستفادة من مجتمع آرتش

- **الشفافية تخلق الاستدامة :لا توجد أسرار في التطوير.**
- **المعرفة المشتركة أقوى من الخبرة الفردية.**
- **الويكي أفضل من أي أداة رسومية معقدة.**

خلاصة القسم 1.8 :المجتمع هو القلب النابض لآرتش لينكس، و Arch Wiki هو دماغه. بدون المجتمع، ما كان لآرتش أن يبقى قويًا ومرنًا لأكثر من عقدين من الزمن. ولأي مستخدم جديد، أول شيء يجب أن يتعلمه هو كيفية قراءة التوثيق، وكيفية السؤال بوضوح، وكيفية مشاركة خبرته مع الآخرين.

1.9آرتش لينكس كأساس لتوزيعات أخرى

إحدى أهم علامات نجاح أي توزيعة هي قدرتها على أن تصبح أساسًا لتوزيعات أخرى تبنى فوقها. بفضل بساطتها، وحزمها الحديثة، ومستودعاتها القوية، أصبح آرتش لينكس قاعدة لعشرات التوزيعات المشتقة.

1.9.1مانجارو لينكس: آرتش للمبتدئين

ظهرت مانجارو عام 2011 من فريق ألماني بهدف جعل آرتش في متناول المبتدئين. فبينما يركز آرتش على التثبيت اليدوي وإدارة النظام بخطوات دقيقة، توفر مانجارو:

- مثبتًا رسوميًا سهلًا.
- إعدادات جاهزة للتعريفات (خاصة لبطاقات .(NVIDIA
- مستودعات خاصة بها تقوم بتجميد الحزم لفترة قصيرة لاختبارها قبل طرحها، مما يوفر استقرارًا أكبر.

تعتبر مانجارو "بوابة" إلى عالم آرتش، حيث تمنح المستخدم تجربة قريبة من آرتش ولكن مع راحة إضافية.

1.9.2إنديفور أو إس: روح المجتمع

تعود خلفية إنديفور أو إس إلى مشروع (Antergos (2002–2019، الذي كان يهدف إلى تقديم آرتش مع مثبت رسومي وتجربة جاهزة. بعد توقفAntergos ، وُلد مشروع EndeavourOSعام 2019 كمبادرة مجتمعية.

تختلف عن مانجارو في أنها تحاول أن تكون أقرب إلى آرتش "الخام" ولكنها توفر:

- مثبتًا رسوميًا.
- خيارات متعددة لبيئات سطح المكتبKDE) ، XFCE، GNOME، إلخ.(
- دعمًا مجتمعيًا قويًا يشبه روح آرتش نفسها.

تُعتبر خيارًا مثاليًا للمستخدم الذي يريد تجربة مطابقة تقريبًا لآرتش، ولكن مع بداية أسهل.

1.9.3حلوان لينكس: الهوية المصرية في عالم آرتش

حلوان لينكس هي توزيعة مبنية على آرتش، أُنشئت بهدف الجمع بين قوة آرتش وهويته البسيطة مع الرغبة في خلق تجربة محلية مميزة.

مميزات حلوان لينكس:

- تعتمد مباشرة على مستودعات آرتش، مع إضافة تحسينات وتجارب محلية.
- واجهات وأدوات تسهل التعامل مع Pacman، مثل أداة .(hpm (Helwan Package Manager

- **تركيز على البساطة وهوية بصرية واضحة، مما يمنحها طابعًا مختلفًا عن مجرد استنساخ آخر لآرتش.**

يعكس حلوان لينكس فكرة أن آرتش ليس مجرد توزيعة، بل هو منصة مفتوحة تسمح لأي مطور أو فريق بإنشاء مشروع جديد فوقها.

1.9.4دروس من التوزيعات المشتقة

لا يُقاس نجاح آرتش بمستخدميه المباشرين فقط، بل بمدى انتشار "أبنائه". مانجارو، وإنديفور أو إس، وحلوان لينكس هي أمثلة على كيفية تكييف آرتش ليناسب شرائح مختلفة من المستخدمين:

- **المبتدئ الذي يريد السهولة.**
- **المستخدم المتوسط الذي يريد المرونة مع بداية أسرع.**
- **المجتمعات المحلية التي تريد هويتها الخاصة.**

خلاصة القسم 1.9 :لم يعد آرتش لينكس مجرد توزيعة قائمة بذاتها؛ بل أصبح نواة لعائلة واسعة من التوزيعات. هذه التوزيعات تثبت قوة الفلسفة الكامنة وراء آرتش وقابليتها للبناء، لتلبية احتياجات جمهور متنوع حول العالم.

1.10حالات الاستخدام

على الرغم من أن آرتش لينكس هو توزيعة للأغراض العامة يمكن لأي شخص تثبيتها، إلا أن طبيعته تجعله أكثر ملاءمة لفئات معينة من المستخدمين. نستعرض هنا أبرز حالات الاستخدام:

1.10.1للمطورين: الوصول إلى أحدث الحزم

- يستخدم آرتش لينكس نموذج التحديث المستمر (Rolling Release)، مما يعني أن الحزم يتم تحديثها باستمرار.
- هذا يجعله بيئة مثالية للمطورين الذين يحتاجون إلى:
  - أحدث إصدارات لغات البرمجةPython) ، Go، Rust، إلخ.(
  - المكتبات وأدوات التطوير الحديثة دون الحاجة إلى انتظار "إصدار جديد" للتوزيعة.
  - القدرة على بناء بيئة تطوير مخصصة بالكامل بناءً على احتياجات مشروعهم.

مثال :مطور يعمل على مشروع ذكاء اصطناعي سيستفيد من توفر أحدث إصدارات TensorFlowأو PyTorchفي مستودعات آرتش أو مستودع .AUR

1.10.2للباحثين والأكاديميين: التحكم الكامل بالبيئة

- غالبًا ما يحتاج الباحثون إلى بيئة تجريبية قابلة للتخصيص بالكامل.
- يوفر آرتش:
  - القدرة على تثبيت نواة مخصصة أو إصدار معين من المكتبات.
  - التحكم الدقيق بالإصدارات، مما يساعد في إعادة إنتاج النتائج العلمية.
  - مرونة استخدام الأدوات من مستودع AURأو حتى بناء الحزم يدويًا.

هذا الأمر مهم بشكل خاص في المجالات العلمية مثل الحوسبة عالية الأداء(HPC) ، والفيزياء، والرياضيات التطبيقية.

1.10.3للمستخدم العادي: بين الفضول والتحدي

- ليس آرتش التوزيعة الأنسب للمبتدئين أو المستخدمين الذين يبحثون عن تجربة "فقط تعمل.(just works) "
- ومع ذلك، هو خيار ممتاز إذا كان المستخدم:
  - يريد تعلم كيفية عمل لينكس من الداخل.
  - يفضل التحكم الكامل على الإعدادات المسبقة.

- **يمتلك الوقت والفضول لخوض رحلة تعليمية عبر التثبيت اليدوي وإدارة النظام.**

قد يجد المستخدم الذي يريد بيئة جاهزة وسريعة دون عناء، توزيعات مثل أوبونتو، أو فيدورا، أو حتى مانجارو أكثر ملاءمة.

خلاصة القسم 1.10 :آرتش لينكس هو الملعب المثالي للمطورين والباحثين الذين يحتاجون إلى الحداثة والمرونة. أما المستخدم العادي فيمكنه الاستفادة من التجربة كرحلة تعليمية، لكنها قد لا تكون الخيار الأفضل لبيئة العمل اليومية ما لم يكن شغوفًا باستكشاف أعماق لينكس.

1.11خاتمة الفصل الأول

بعد هذه الجولة في عالم آرتش لينكس، يمكننا أن نستنتج أن آرتش ليس مجرد توزيعة أخرى بين عشرات التوزيعات؛ بل هو منهج وفلسفة شاملة.

- **فلسفته مبنية على البساطة، الشفافية، والتحكم الكامل.**
- **التعامل معه يجعلك أقرب إلى نواة لينكس ويساعدك على فهم تركيبته الداخلية، بدلًا من الاكتفاء بالقشرة الخارجية التي توفرها التوزيعات الأخرى.**
- **كل خطوة في استخدام آرتش هي رحلة تعليمية: من التثبيت اليدوي إلى إدارة الحزم عبر pacman، وحتى تخصيص سطح المكتب حسب رغبتك.**

يمكن القول إن آرتش لينكس أشبه بـ "ورشة عمل تعليمية "مفتوحة للمستخدم:

- **إذا كنت مطورًا، ستجد أحدث الحزم والأدوات في متناول يدك.**
- **إذا كنت باحثًا، ستكون لديك القدرة على التحكم في كل تفصيلة صغيرة وكبيرة داخل بيئتك.**
- **وإذا كنت مستخدمًا عاديًا شغوفًا بالتعلم، سيمنحك آرتش فرصة لفهم لينكس بعمق لا توفره أي توزيعة أخرى.**

وبالتالي، يصبح تعلم آرتش بوابة لفهم أعمق للينكس نفسه، ويمنحك ثقة أكبر في التعامل مع أنظمة التشغيل مفتوحة المصدر.

بانتهاء هذا الفصل التمهيدي، نكون قد وضعنا الأساس النظري لفهم آرتش لينكس. الآن، حان الوقت للانتقال إلى الفصل الثاني: إعداد بيئة بناءArchiso ، حيث سنبدأ الجانب العملي ونكتشف كيف يمكننا إعداد بيئة العمل لبناء توزيعة مخصصة بدءًا من آرتش.

الفصل الثاني: إعداد بيئة البناء باستخدامArchiso

2.1مقدمة إلىArchiso

ما هوArchiso ؟ Archisoهو الإطار الرسمي الذي طوّره مجتمع Arch Linux وصانوه لبناء صور ISO قابلة للإقلاع (Live ISO images).تخدم هذه الصور أغراضًا متنوعة، بما في ذلك توزيعة النظام، صيانة الأنظمة، والاختبار. بشكل أساسي، هو "باني صور ISO"الخاص بآرتش، ويسمح لك بـ:

- **بناء نسخة افتراضية من آرتش لينكس مطابقة للإصدار الرسمي.**
- **أو، الذهاب أبعد من ذلك وإنشاء توزيعة مخصصة بالكامل بهوية جديدة، مثل حلوان لينكس.**

يُبنى Archiso على مبدأ البساطة: فبدلاً من الاعتماد على أدوات معقدة أو أنظمة خارجية، يتكون من مجموعة من السكربتات وملفات الإعدادات التي تمكنك من توليد نظام تشغيل قابل للإقلاع بصيغة.ISO

لماذا نستخدمArchiso ؟ لفهم أهميته، دعنا نسأل: "ما الذي يحدد وجود توزيعة لينكس؟" الإجابة تكمن في ملف ISO متاح بسهولة، يمكن للمستخدمين تنزيله وحرقه على محرك أقراص USB والقيام بالإقلاع منه Archiso .هي الأداة التي تمكنك من إنشاء هذا الملف بنفسك.

تتضمن الأسباب الرئيسية لاستخدامه:

- **التحكم الكامل :أنت من يحدد الحزم، والإعدادات، وبيئة سطح المكتب.**
- **توزيعة شخصية :قم ببناء نظامك الشخصي وشاركه مع الآخرين.**
- **أغراض الصيانة :أنشئ ملف ISO مخصصًا لفريقك، مزودًا بأدوات متخصصة في الشبكات أو الأمان.**
- **الاستقرار والاختبار :اختبر نسخة من نظامك في بيئة افتراضية مثل VirtualBox قبل توزيعها على نطاق أوسع.**

يمكن القول إن Archiso هو "المسبك" الذي تخرج منه جميع المشاريع المبنية على آرتش.

الفرق بين بناء تثبيت آرتش بسيط وتخصيص توزيعة مثل حلوان لينكس

لتوضيح الأمر أكثر، دعنا نقارن بين سيناريوهين:

العنصر	بناء تثبيت آرتش بسيط	تخصيص توزيعة مثل حلوان لينكس
الهدف	نسخة طبق الأصل من آرتش الرسمي	نسخة مخصصة بهوية جديدة
الحزم المثبتة	الحزم الأساسية فقط	بيئات سطح المكتب + أدوات خاصة بحلوان
الهوية البصرية	شعار آرتش، خلفيات افتراضية	شعار حلوان، سمات مخصصة
الجمهور المستهدف	مستخدمو آرتش المتمرسون	المستخدمون الجدد + المحترفون
حجم ملفISO	معتدل (700 – 900 ميجابايت(	متغير: خفيف جدًا (Fluxbox) إلى ثقيل نسبيًا(Cinnamon)

تُبرز هذه المقارنة أن Archiso ليس مجرد أداة نسخ، بل هو آلية قوية لصناعة "تجربة مستخدم" مميزة.

المتطلبات الأساسية للبناء باستخدامArchiso

قبل الشروع في أي عملية بناء، تأكد من استيفاء المتطلبات التالية:

- **أجهزة مناسبة:**
  - **معالج حديث (يفضل x86_64، مع دعم virtualizationإذا كنت ستختبر على .(QEMU/VirtualBox**
  - **ذاكرة وصول عشوائي (RAM) لا تقل عن 4جيجابايت)ويوصى بـ 8جيجابايت.(**
  - **مساحة تخزين 20 :جيجابايت أو أكثر (تتطلب عملية البناء ذاكرة مؤقتة ودلائل مؤقتة.(**
  - **يوصى بشدة بقرص SSDلتسريع عمليات الضغط وفك الضغط.**
- **نظام آرتش لينكس أو نظام مشتق منه:**
  - **يجب أن يكون لديك نظام آرتش لينكس مثبت، أو توزيعة مبنية عليه (مثل حلوان لينكس.(**
  - **السبب :يعتمد Archiso على مكتبات وأدوات محددة من بيئة آرتش.**
- **اتصال إنترنت جيد:**
  - **تتضمن عملية البناء تنزيل مئات الحزم من المستودعات.**
  - **قد يؤدي الاتصال الضعيف إلى أخطاء في التوقيع.**
- **صلاحيات الجذر:**
  - **تتطلب عملية البناء تعديل ملفات أساسية وتثبيت حزم.**
  - **يفضل استخدام sudoبدلًا من تسجيل الدخول كـ rootمباشرةً.**

ملاحظة تاريخية

Archisoليس تطورًا حديثًا.

- **كانت أصوله بسيطة جدًا: سكريبت بسيط لبناء ملف ISO الرسمي لآرتش.**
- **بمرور الوقت، اكتشف المستخدمون إمكاناته لإنشاء نسخ مخصصة خاصة بهم.**
- **اليوم، تعود أصول معظم التوزيعات المبنية على آرتش (مثل Manjaro، EndeavourOS، Artix، و حلوان لينكس (إلى Archiso.**

مثال عملي لتوضيح المفهوم

تخيل هذا السيناريو:

- **أنت تعمل على تثبيت آرتش عادي.**
- **قررت بناء نسخة تجريبية خفيفة الوزن تُسمى .Helwan Light**
- **ببساطة، تقوم بنسخ ملفات الإعدادات منArchiso ، وتعدّل ملف packages.x86_64، وتزيل بيئات سطح المكتب الثقيلة مثل GNOMEأو .KDE**
- **في غضون ساعة أو ساعتين، سيكون لديك ملف ISO بحجم 600 ميجابايت يُقلع إلى بيئة Fluxboxخفيفة جدًا.**

هذا يوضح كيف يبسّط Archiso العملية بشكل كبير ويوفر بيئة ملموسة للتجريب.

خاتمة (مقدمة الفصل(

- **Archisoهو المحرك الأساسي لبناء أي توزيعة مبنية على آرتش.**
- **إنه يمكنك من إعادة بناء نسخة بسيطة أو الابتكار بتوزيعة جديدة تحمل هويتك الفريدة.**
- **المتطلبات الأساسية ليست صعبة للغاية ولكنها تتطلب أجهزة مناسبة ونظام آرتش جاهز.**
- **ستوجهك الأقسام القادمة (2.2 – 2.9) خطوة بخطوة، بدءًا من تثبيت الأدوات الأساسية وصولًا إلى اختبار توزيعتك في VirtualBox.**

2.2مقدمة إلىArchiso

تتمحور عملية بناء توزيعة مبنية على آرتش لينكس باستخدام أداة archisoحول دليل /releng، والذي يُعد نواة عملية البناء. يحتوي هذا الدليل على جميع المكونات التي تُحدد صورة ISO النهائية، بدءًا من نظام الملفات الجذر (/airootfs) وقوائم الحزم المطلوبة وصولًا إلى ملفات الإقلاع لكل من أنظمة BIOSو .UEFI

يوضح هذا المستند وظيفة كل ملف ومجلد داخل /releng، مع شرح كيفية دمج هذه العناصر لإنشاء بيئة حية جاهزة للاستخدام أو التثبيت. فهم هذا الهيكل يمنح مطور التوزيعة القدرة على تخصيص كل جزء بدقة، من الحزم المضمنة إلى واجهة المستخدم عند الإقلاع الأول.

تثبيت أداة archisoأداة archisoهي المسؤولة عن بناء صورة .ISO إنها متاحة في مستودع extraالرسمي لآرتش لينكس ويمكن تثبيتها بالأمر التالي:

Bash

```
sudo pacman -S archiso
```

ملاحظة :يُنصح بالقيام بعملية البناء داخل بيئة آرتش نظيفة (أو على الأقل داخل chrootأو (VMلتجنب التعارضات مع النظام المضيف.

الحصول على ملف إعدادات releng تأتي أداة archisoمع ملفات إعدادات معدة مسبقًا(profiles) ، بما في ذلك releng) نفس الملف الذي يستخدمه آرتش لينكس لإنتاج ملف ISO الرسمي .(لنسخ ملف الإعدادات إلى دليل العمل الخاص بك:

Bash

```
cp -r /usr/share/archiso/configs/releng/ ~/archlive/
```

الآن، لديك نسخة محلية من ملف الإعدادات في /archlive/~يمكنك تعديلها بحرية دون التأثير على النسخة الأصلية.

```
 archlive /
├──  airootfs
|   ├──  etc
|   ├──  root
|   └──  usr
├──  bootstrap_packages.x86_64
├──  efiboot
├──  grub
├──  packages.x86_64
```

```
├──  pacman.conf
├──  profiledef.sh
└──  syslinux
    ├──  archiso_head.cfg
    ├──  archiso_pxe-linux.cfg
    ├──  archiso_pxe.cfg
    ├──  archiso_sys-linux.cfg
    ├──  archiso_sys.cfg
    ├──  archiso_tail.cfg
    ├──  splash.png
    └──  syslinux.cfg
```

يُقدم هذا المخطط الشجري نظرة عامة على تخطيط ملف إعدادات .archisoفي الأقسام التالية، سيتم شرح كل دليل وملف بالتفصيل.

releng/

هو الدليل الأساسي لملف إعدادات .archisoكل ما بداخله يُحدد:

- **الحزم التي سيتم تثبيتها في ملف .ISO**
- **إعدادات الإقلاع .(EFI / BIOS)**
- **ملفات تخصيص المستخدم و.root**
- **إعدادات .Pacman**
- **ملفات محمل الإقلاع .(grub/syslinux)**

airootfs/

هذا هو الجزء الأكثر أهمية، ويمثل نظام الملفات الجذر (rootfs) الذي سيتم بناؤه داخل ملف الـ .ISO أي ملف يتم وضعه هنا سيظهر تحت /بعد إقلاع النظام من الـ.ISO

- **→ /etc إعدادات النظام :يحتوي على ملفات الإعداد مثل shadow، passwd، hostname، issue، إلخ. أي تغييرات هنا تنعكس في النظام الحي.(live system)**
- **→ /root دليل المستخدم root داخل البيئة الحية. هنا يمكنك وضع سكربتات التخصيص مثل customize_airootfs.shأو ملفات الإعداد الشخصية (dotfiles) مثل bashrc.، .zshrc.**
- **→ /usr ملفات ومكتبات إضافية للمستخدم. مثل إضافات لـ /binو /lib، أو أي أدوات تريد تثبيتها مسبقًا.**

bootstrap_packages.x86_64

قائمة بالحزم الأساسية اللازمة لبناء بيئة ISO الأولية (بيئة .(bootstrap هذه الحزم لا تُضمّن في النظام النهائي، ولكنها تُستخدم أثناء عملية البناء للسماح لـ archisoبالدخول إلى بيئة معزولة (chroot) ومتابعة البناء.

efiboot/

يحتوي على ملفات محمل الإقلاع) EFI إما systemd-bootأو GRUB لـ .(UEFI هذا المجلد يحمل ملفات efi.والإعدادات الخاصة بمدخل الإقلاع.UEFI

grub/

إعدادات محمل الإقلاع) GRUB إذا اخترت استخدامه). يحتوي على ملفات الإقلاع لقائمةGRUB ، والخلفية، وقوالب الإعدادات.

packages.x86_64

هذه هي قائمة الحزم التي ستكون متاحة داخل النظام الحي بعد الإقلاع. إنها أهم ملف لتخصيص ملف الـ ISO الخاص بك. هنا، ستُدرج:

- **الأدوات الأساسية (مثل bash، vim، .(nano**
- **بيئات سطح المكتب أو مديري النوافذ.(window managers)**
- **أدوات الشبكة والصيانة.**

pacman.conf

ملف إعدادات مدير الحزم pacman داخل النظام الحي. يمكنك استخدامه لتحديد:

- **المستودعات الرسمية.**
- **مستودعات إضافية (إضافية/مخصصة.(**
- **التحقق من التوقيعات وخيارات أخرى.**

profiledef.sh

هو "عقل" ملف الإعدادات، حيث يُعرّف خصائصه الأساسية. يُحدد:

- **اسم التوزيعة iso_name)، .(iso_label**
- **الإصدار .(iso_version)**
- **النواة (kernel)التي سيتم استخدامها.**
- **إعدادات وضع الإقلاع .(UEFI/BIOS)**
- **دلائل البناء والمراحل المؤقتة.(build and staging directories)**
- **يتبع archisoهذا الملف خطوة بخطوة لبناء ملف الـISO.**

syslinux/

يحتوي على ملفات محمل الإقلاعSyslinux ، والذي يُستخدم للإقلاع عبر .BIOS/Legacy

- **← archiso_head.cfg بداية إعداد قائمة الإقلاع (العنوان، المهلة الزمنية.(**
- **← archiso_pxe-linux.cfg & archiso_pxe.cfg إعدادات الإقلاع عبر) PXEالإقلاع عبر الشبكة.(**
- **← archiso_sys-linux.cfg & archiso_sys.cfg إعدادات الإقلاع القياسية من ملف الـISO.**
- **← archiso_tail.cfg نهاية إعداد قائمة الإقلاع.**
- **← splash.png خلفية رسومية لشاشة.Syslinux**
- **← syslinux.cfg الملف الرئيسي الذي يضم ملفات الإعدادات الأخرىhead) ، tail، sys، (pxeويُعرّف قائمة الإقلاع.**

ملخص:

- **airootfs/ = نظام الملفات الجذر الذي سيتم بناؤه داخل ملف الـISO.**
- **packages.x86_64 = الحزم التي ستكون في النظام الحي.(live system)**
- **bootstrap_packages.x86_64 = الحزم المطلوبة لعملية البناء فقط.**
- **pacman.conf = إعدادات مدير الحزم.**
- **profiledef.sh = خصائص التوزيعة والإصدار.**
- **efiboot/ + grub/ + syslinux/ = ملفات الإقلاع لكل وضع.**

2.3ملفات الإعداد الأساسية(Core Configuration Files)

يمثل هذا الفصل حجر الزاوية في رحلتنا لتخصيص توزيعة آرتش لينكس. هنا، سنتعمق في قلب عملية البناء، مستكشفين الملفات التي لا تحدد فقط محتوى ملف الـ ISO النهائي، بل تتحكم أيضًا في كيفية بناء هذا الملف.

فهم هذه الملفات الأربعةprofiledef.sh — ، packages.x86_64، bootstrap_packages.x86_64، و — pacman.conf هو مفتاح تحقيق تحكم كامل بمشروعك، مما يمكنك من بناء توزيعة فريدة حقًا.

نحن لا نتحدث فقط عن قوائم الحزم؛ هذه الملفات تمثل "وصفة" متكاملة يتبعها archisoخطوة بخطوة لتهيئة النظام. كل سطر في هذه الملفات له غرض محدد، وكل خيار يترجم إلى سلوك معين أثناء عملية البناء أو داخل النظام الحي بعد الإقلاع.

2.3.1 profiledef.sh:العقل المدبر وراء توزيعتك المخصصة

هذا الملف هو جوهر أي ملف إعدادات لـ archiso. إنه سكريبت Bash يحتوي على جميع التعريفات والإعدادات التي تحدد هوية صورة الـ ISO الخاصة بك، وسلوك إقلاعها، وعملية بنائها. اعتبره "قائمة المهام" التي يتبعها archisoلإنتاج نظامك الحي.

- #!/usr/bin/env bash: هذا هو سطر "shebang" المعروف، والذي يوجه النظام لتنفيذ هذا السكريبت باستخدام مترجم Bash.
- # shellcheck disable=SC2034: هذا تعليق خاص بأداة shellcheck) أداة تحليل سكريبتات .(Bash هنا، نطلب من shellcheckتجاهل تحذير معين يتعلق بالمتغيرات غير المستخدمة(SC2034) ، حيث قد تكون بعض المتغيرات مخصصة لأغراض محددة أو مستخدمة داخليًا بواسطة archiso.

بيانات التعريف الخاصة بـISO (ISO Metadata)

هذه المتغيرات تُعرّف هوية وخصائص صورة الـ ISO النهائية.

- iso_name="Helwan-Linux-stable": اسم ملف الـ ISO الذي سيتم إنشاؤه. في مثالنا، سيكون -Helwan-Linux stable-x86_64.iso) عادةً، يضيف archisoتلقائيًا بنية المعالج.(
- iso_label="Helwan-Linux-stable-v1.1": تسمية الـ ISO. هذا هو النص الذي يظهر عند تحميل الـ ISO أو الإقلاع منه (على سبيل المثال، في قائمة مدير الإقلاع). يجب أن يكون موجزًا (عادةً بحد أقصى 32 حرفًا، ولكن 16 حرفًا هو الأفضل لتوافق أوسع.(
- iso_publisher="helwanlinux <helwanlinux@gmail.com>": معلومات الناشر. تتضمن اسم الجهة المسؤولة عن التوزيعة ونقطة اتصال (عنوان بريد إلكتروني في هذا المثال.(
- iso_application="Helwan Linux Live/Rescue DVD": وصف موجز لغرض الـ ISO. يوضح استخدامه المقصود، مثل "Live/Rescue DVD" أو."System Installer"
- iso_version="v1.1": رقم إصدار التوزيعة. هذا مهم لتتبع التحديثات وتحديد الإصدارات المحددة.

إعدادات البناء الأساسية(Core Build Settings)

تُعرّف هذه المتغيرات بنية وعناصر عملية البناء الأساسية.

- **install_dir="arch":دليل التثبيت داخل الـ .ISO عندما يقوم المستخدم بتحميل الـISO ، سيتم وضع ملفات النظام في هذا المسار. الافتراضي لـ Arch Linux هو .arch**
- **buildmodes=('iso'):أوضاع البناء المتاحة.**
  - **iso:يشير إلى أننا سنبني صورة ISO قابلة للإقلاع.**
  - **قد تشمل الأوضاع الأخرى (tarلإنشاء أرشيف لنظام الملفات) أو (vendorلإنشاء صور مخصصة لأجهزة معينة)، ولكن isoهو الأكثر شيوعًا.**
- **arch="x86_64":بنية المعالج المستهدفة. هنا، x86_64تعني أن الصورة مصممة لأنظمة 64 بت.**
- **pacman_conf="pacman.conf":اسم ملف إعدادات .Pacman يشير هذا إلى ملف pacman.confالموجود في نفس دليل ملف الإعدادات، والذي سيُستخدم لتهيئة مدير الحزم داخل النظام الحي.**
- **airootfs_image_type="squashfs":نوع ضغط نظام الملفات الجذر.**
  - **squashfs:هذا هو النوع القياسي للأنظمة الحية(Live Systems) ، ويوفر ضغطًا ممتازًا وسرعة قراءة.**
  - **قد تشمل الخيارات الأخرى ext4، ولكن squashfsهو الأكثر استخدامًا لصور الـISO.**

خيارات وأدوات الضغط(Compression Options and Tools)

تحدد هذه المتغيرات كيفية ضغط نظام الملفات الجذر (airootfs) والملفات الأخرى.

- **airootfs_image_tool_options=('-comp' 'xz' '-Xbcj' 'x86' '-b' '1M' '-Xdict-size' '1M'):خيارات لأداة ضغط .squashfs**
  - **'xz' '-comp':يستخدم خوارزمية ضغط xz، والتي توفر نسبة ضغط عالية جدًا مقابل وقت معالجة أطول.**
  - **'x86' '-Xbcj':يحدد فلترًا خاصًا (x86 Code Branch) لتطبيق الضغط على شيفرة بنيةx86 ، مما يحسن نسبة الضغط.**
  - **'1M' '-b':حجم الكتلة (block size) الذي ستستخدمه الأداة. حجم أكبر يعني عادةً ضغطًا أفضل ولكنه يتطلب ذاكرة أكبر أثناء الضغط.**
  - **'1M' '-Xdict-size':حجم قاموس الضغط .(dictionary size) قاموس أكبر يؤدي بشكل عام إلى ضغط أفضل.**
- **bootstrap_tarball_compression=('zstd' '-c' '-T0' '--auto-threads=logical' '--long' '-19'):خيارات الضغط للأرشيف المستخدم لبناء البيئة الأولية.**
  - **هنا، نستخدم zstd، وهو ضاغط سريع وفعال جدًا.**
    - **c-:يكتب المخرجات إلى المخرج القياسي.(stdout)**

- **T0:-يستخدم جميع الأنوية المعالجة المتاحة (لتحقيق أقصى سرعة.)**
- **auto-threads=logical:--يضبط عدد الخيوط تلقائيًا بناءً على عدد الأنوية المنطقية.**
- **long:--يُمكّن وضع"long" ، والذي يزيد من فعالية الضغط (مفيد للملفات الكبيرة.)**
- **19:-مستوى الضغط (19 هو الأعلى، مما يعني أقصى ضغط مقابل وقت أطول.)**

أوضاع الإقلاع(Boot Modes)

هذا المتغير هو أحد أهم المتغيرات، لأنه يحدد كيف يمكن إقلاع ملف الـISO.

bootmodes=('bios.syslinux.mbr'

'bios.syslinux.eltorito'

'uefi-ia32.systemd-boot.esp' 'uefi-x64.systemd-boot.esp'

'uefi-ia32.systemd-boot.eltorito' 'uefi-x64.systemd-boot.eltorito')

يعكس هذا مدى تعقيد وتوافق .archisoدعنا نحلل هذه السلاسل:

- **bios.syslinux.mbr:**
  - **bios:يشير إلى وضع إقلاع (BIOSالقديم.)**
  - **syslinux:يستخدم Syslinux كمحمل إقلاع.**
  - **mbr:يستخدم Master Boot Record (MBR) لإنشاء قطاع إقلاع قابل للتنفيذ على القرص، مما يسمح بالإقلاع من الأجهزة القديمة.**
- **bios.syslinux.eltorito:**
  - **bios:وضع .BIOS**
  - **syslinux:محمل إقلاع.Syslinux**
  - **eltorito:يستخدم معيار El Torito لإنشاء قرص CD/DVD قابل للإقلاع. هذا هو التنسيق القياسي لصور الـ ISOالقابلة للإقلاع.**
- **uefi-ia32.systemd-boot.esp:**
  - **uefi:يشير إلى وضع إقلاع .UEFI (Unified Extensible Firmware Interface)**
  - **ia32:يشير إلى دعم المعالجات 32بت .(x86)**
  - **systemd-boot:يستخدم (systemd-bootالمعروف سابقًا بـ gummiboot)كمحمل إقلاع.UEFI**
  - **esp:يشير إلى هيكل الملفات داخل الـ ISO الذي يحاكي قسم نظام.(EFI System Partition - ESP) EFI**

- **uefi-x64.systemd-boot.esp:**
  - **uefi:وضع UEFI.**
  - **x64:يدعم معالجات 64بت .(x86_64)**
  - **systemd-boot:محمل الإقلاع systemd-boot.**
  - **esp:هيكل ملفات ESP.**
- **uefi-ia32.systemd-boot.eltorito:**
  - **uefi:وضع UEFI.**
  - **ia32:يدعم 32بت.**
  - **systemd-boot:محمل الإقلاع systemd-boot.**
  - **eltorito:يستخدم معيار El Torito لبناء صورة ISO قابلة للإقلاع لـ UEFI.**
- **uefi-x64.systemd-boot.eltorito:**
  - **uefi:وضع UEFI.**
  - **x64:يدعم 64بت.**
  - **systemd-boot:محمل الإقلاع systemd-boot.**
  - **eltorito:معيار El Torito لإنشاء صورة ISO قابلة للإقلاع لـ UEFI.**

تطبيق عملي :هذا يعني أن الصورة الناتجة ستكون قادرة على الإقلاع من معظم الأجهزة الحديثة (UEFI) والقديمة(BIOS) ، باستخدام محملات الإقلاع الافتراضية لـ Arch Linux (systemd-boot) لـ UEFI و Syslinux لـ.(BIOS

أذونات الملفات(File Permissions)

يحدد هذا القسم الأذونات للملفات والأدلة الهامة داخل نظام الملفات الجذر (airootfs) لضمان الأمان والتشغيل السليم.

```
file_permissions=(
 ["/etc/shadow"]="0:0:400"
 ["/root"]="0:0:750"
 ["/root/.automated_script.sh"]="0:0:755"
 ["/root/.gnupg"]="0:0:700"
 ["/usr/local/bin/choose-mirror"]="0:0:755"
 ["/usr/local/bin/Installation_guide"]="0:0:755"
```

["/usr/local/bin/livecd-sound"]="0:0:755"

["/etc/polkit-1/rules.d"]="0:0:750"

["/etc/sudoers.d"]="0:0:750"

)

هنا، يُستخدم التنسيق ."owner:group:permissions"=["file_path"]

- owner:group: معرّف المستخدم (UID) ومعرّف المجموعة (GID). 0:0 تعني عادةً المستخدم root والمجموعة root.
- permissions: الأذونات بتنسيق ثماني (octal).
  - 400: للقراءة فقط للمالك (r--------).
  - 750: للقراءة والكتابة والتنفيذ للمالك (rwxr-x---) ؛ للقراءة والتنفيذ للمجموعة.
  - 755: للقراءة والكتابة والتنفيذ للمالك (rwxr-xr-x) ؛ للقراءة والتنفيذ للآخرين.
  - 700: للتنفيذ فقط للمالك (rwx------).

أهمية هذه الأذونات:

- /etc/shadow: يحتوي على كلمات المرور المشفرة. يجب أن يكون مملوكًا لـ root وقابلاً للقراءة بواسطة root فقط (400).
- /root: دليل المستخدم root. 750 يعني أن root يمكنه القراءة والكتابة والتنفيذ، بينما يمكن للمجموعة القراءة والتنفيذ فقط (مفيد إذا تم إنشاء مجموعات محددة).
- /root/.automated_script.sh: إذا كان لديك سكربتات مؤتمتة، فيجب أن تكون قابلة للتنفيذ (755).
- /root/.gnupg: للمجلدات التي تحتوي على مفاتيح GPG ، من الأفضل تقييد الوصول إلى root فقط (700).
- /usr/local/bin/: هذا المسار هو الموقع القياسي لإضافة السكربتات المخصصة. تعيين أذونات 755 يجعلها قابلة للتنفيذ من قبل الجميع.
- /etc/polkit-1/rules.d و /etc/sudoers.d: تحتوي هذه المجلدات على قواعد تصعيد الامتيازات. تقييد الوصول إليها (750) يمنع المستخدمين غير المصرح لهم من تعديلها.

ملخص الفصل

ملف profiledef.sh هو كنز دفين للتحكم في عملية بناء الـ ISO. لقد رأينا كيف يحدد هذا الملف:

- هوية التوزيعة (الاسم، الإصدار، الناشر).
- بنية النظام (المعالج، دليل التثبيت، نوع صورة النظام).
- ميزات الإقلاع (BIOS ، UEFI، Syslinux، systemd-boot).

- **إعدادات الأمان (أذونات الملفات.(**
- **أدوات الضغط لتحسين حجم الـ ISO وكفاءة البناء.**

فهمك العميق لهذا الملف سيمكنك من تخصيص توزيعتك بما يتجاوز مجرد قائمة الحزم، ليصل إلى سلوك النظام الأولي. في الفصول اللاحقة، سنتوسع في هذه التعريفات بينما نتعمق في مجلدات /airootfsوملفات الإقلاع.

2.3.2 packages.x86_64:قائمة حزم النظام الحي

إذا كان profiledef.shهو المخطط، فإن packages.x86_64هو قائمة المكونات الفعلية للنظام. هذا الملف هو قائمة نصية بسيطة بالحزم التي سيقوم archisoبتثبيتها داخل النظام الحي.(Live System)

فلسفة الاختيار

عند بناء توزيعة مخصصة، فكّر بعناية في الحزم التي تُدرجها. هناك توازن بين توفير بيئة غنية بالميزات والحفاظ على حجم ملف الـ ISOصغيرًا.

مثال للمحتوى

```
# Core System

base

linux

linux-firmware

nano

vim


# Desktop Environment (Example: GNOME)

xorg-server

gnome

gnome-extra


# Networking and Connectivity

networkmanager

dhcpcd

wpa_supplicant
```

 # System Utilities

htop

neofetch

pacman-contrib

نصائح لتخصيص القائمة:

- **ابدأ بـ base وlinux: هذه الحزم ضرورية لتوفير نظام قابل للإقلاع.**
- **أضف الأدوات الأساسية :أدوات مثل nano، vim، و gitغالبًا ما تكون ضرورية للمستخدمين.**
- **حدد بيئة سطح المكتب الخاصة بك :اختر بيئة سطح مكتب (مثلGNOME ، KDE Plasma، XFCE)أو مدير نوافذ (مثل i3، Sway)مع جميع الاعتماديات الضرورية.**
- **ضع في اعتبارك الأجهزة :إذا كانت توزيعتك تستهدف أجهزة معينة، فكّر في تضمين التعريفات الخاصة (مثل بطاقات Wi-Fi أو بطاقات الرسوميات.(**
- **إدارة الحزم :يستخدم archisoأداة pacman، ولكن يمكنك إضافة أدوات مساعدة مثل yayإذا رغبت في ذلك.**

2.3.3 bootstrap_packages.x86_64:أساسيات عملية البناء

على عكس packages.x86_64، فإن الحزم المدرجة في bootstrap_packages.x86_64ليست جزءًا من النظام الحي النهائي. بدلاً من ذلك، هي الحزم التي يحتاجها archisoنفسه لإنشاء بيئة بناء معزولة (chroot) وتثبيت الحزم الأخرى. هذه الحزم ضرورية للتنفيذ الفعال لعملية التجميع.

مثال لمحتوى bootstrap_packages.x86_64

)مثال("Helwan Linux" :

arch-install-scripts

archiso

base

base-devel

linux

linux-firmware

شرح هذه الحزم:

- **arch-install-scripts:توفر هذه الحزمة أدوات أساسية لتثبيتArch Linux ، وأهمها الأداة .arch-chrootتسمح هذه الأداة لـ archisoبالدخول إلى بيئة النظام الذي يتم بناؤه لإجراء عمليات التثبيت والتعديلات.**
- **archiso:هذه هي الأداة نفسها! يجب أن تكون متاحة داخل بيئة البناء.**
- **base:توفر الحد الأدنى من الحزم الضرورية لعمل نظامArch Linux ، بما في ذلك systemdوالأدوات الأساسية الأخرى.**
- **base-devel:تتضمن أدوات تطوير البرمجيات الأساسية (مثل gcc، makeالتي قد يحتاجها archisoلبناء مكونات معينة (على الرغم من أن هذا أقل شيوعًا في معظم عمليات بناء الـISO).**
- **linuxو linux-firmware:النواة (kernel) وبرامجها الثابتة المرتبطة بها، وهي ضرورية لبناء بيئة تشغيل أساسية.**

متى يتم تعديل هذا الملف:

في معظم السيناريوهات، لن تحتاج إلى تعديل .bootstrap_packages.x86_64القائمة الافتراضية كافية لبناء معظم توزيعات Arch Linux.ستحتاج فقط إلى تعديله إذا كنت تقوم ببناء متخصص للغاية أو تواجه مشاكل محددة داخل بيئة البناء نفسها.

2.3.4 ملف :pacman.conf مركز التحكم لعالمك المخصص

ملف pacman.confهو محور التحكم المركزي لـpacman ، وهو مدير الحزم الذي يشكل العمود الفقري لإدارة البرمجيات في Arch Linux.عندما تقوم ببناء صورة ISO باستخدامarchiso ، فإن هذا الملف يحدد كيفية جلب الحزم وتثبيتها، سواء داخل بيئة النظام الحي (live system) أو خلال مرحلة البناء الأولية .(bootstrap) دعنا نفصّل هذا الملف سطرًا بسطر، مع التركيز على الخيارات الأكثر أهمية لك كمطور توزيعة:

قسم ([options]الإعدادات العامة(

يعمل هذا القسم كلوحة تحكم شاملة لـ.pacman

المسارات:(Paths)

```
#RootDir = /

#DBPath = /var/lib/pacman/

#CacheDir = /var/cache/pacman/pkg/

#LogFile = /var/log/pacman.log

#GPGDir = /etc/pacman.d/gnupg/

#HookDir = /etc/pacman.d/hooks/
```

تحدد هذه المسارات الأماكن التي يخزّن فيها pacman قواعد بياناته(DBPath) ، حيث يحتفظ بالحزم التي تم تنزيلها(CacheDir) ، حيث يكتب سجلات العمليات(LogFile) ، وحيث يبحث عن مفاتيح (GPGDir) GPG والأدوات المساعدة.(HookDir)

- في سياق :archiso عندما يعمل pacman داخل بيئة chroot أثناء عملية البناء، تشير هذه المسارات عادةً إلى مواقع مؤقتة داخل دليل البناء. على سبيل المثال، قد يكون CacheDirموجودًا داخل مجلد مؤقت لمنع تلويث ذاكرة التخزين المؤقت للنظام المضيف.
- للتخصيص :نادرًا ما ستحتاج إلى تغيير هذه المسارات عند بناء صورةISO ، ولكن فهمها ضروري لفهم كيفية عمل pacmanداخليًا.

IgnorePkg / IgnoreGroup: وHoldPkg

```
HoldPkg = pacman glibc

IgnorePkg =

IgnoreGroup =
```

- :HoldPkgيمنع هذا الخيار الحزم المحددة من الترقية التلقائية. هذا أمر حاسم للحفاظ على استقرار النظام، خاصةً للحزم الأساسية مثل pacman و glibc نفسها.
- :IgnorePkgقائمة بالحزم التي لا تريد أبدًا أن يقوم pacman بتحديثها.

- **IgnoreGroup: قائمة بمجموعات الحزم التي لا تريد تحديثها.**
- **في سياق archiso: يُعد الحفاظ على pacman و glibc في HoldPkg ممارسة جيدة لضمان عدم تعطل عملية البناء بسبب تغييرات غير متوقعة في هذه الحزم الحيوية.**

Architecture = auto:

يحدد هذا الخيار بنية النظام المستهدفة. تعني auto أن pacman سيحاول اكتشاف البنية تلقائيًا (مثل.(x86_64

- **في سياق archiso: بما أنك تبني صورة ISO لبنية محددة، يمكنك تعيينها صراحةً هنا (مثلاً Architecture = x86_64)، لكن auto تعمل جيدًا بشكل عام.**

ParallelDownloads = 5:

يحدد هذا الخيار عدد الحزم التي يمكن لـ pacman تنزيلها بشكل متزامن. يمكن أن تؤدي زيادة هذا العدد (مثلاً إلى 5 أو 10) إلى تسريع عملية التنزيل بشكل كبير، خاصةً مع اتصال إنترنت جيد.

- **في سياق archiso: سرعة التنزيل الأسرع تعني بناء أسرع لـISO ، مما يجعل هذا خيارًا مفيدًا للغاية.**

SigLevel و LocalFileSigLevel:

SigLevel = Required DatabaseOptional

LocalFileSigLevel = Optional

تحدد هذه الإعدادات مستوى التحقق من التوقيعات الرقمية للحزم.

- **SigLevel = Required DatabaseOptional: هذا يعني أن pacman سيطلب توقيعًا صالحًا (Required) للحزم التي يتم تنزيلها من المستودعات البعيدة، ولكنه سيقبل الحزم ذات التوقيعات الاختيارية (DatabaseOptional) أو حتى الحزم غير الموقعة من قواعد بيانات المستودعات.**
- **LocalFileSigLevel = Optional: يشير هذا إلى أن الحزم المثبتة من الملفات المحلية (مثل تلك المنسوخة يدويًا) يمكن أن تكون غير موقعة أو موقعة بشكل اختياري.**
- **في سياق archiso: يضمن هذا الإعداد مستوى معينًا من الأمان عند جلب الحزم مع الحفاظ على مرونة كافية لعدم تعطيل البناء في حالة وجود مشكلة مؤقتة في التوقيع.**

قسم ([repositories]مصادر الحزم(

هنا تحدد أين سيبحث pacman عن الحزم. الترتيب هنا حاسم، حيث سيستخدم pacman أول مستودع يحتوي على الحزمة المطلوبة.

المستودعات الرسمية:

[core]

Include = /etc/pacman.d/mirrorlist

[extra]

Include = /etc/pacman.d/mirrorlist

#[multilib]

#Include = /etc/pacman.d/mirrorlist

- **[extra], [core]:هذه هي المستودعات الأساسية في Arch Linux.**
- **Include = /etc/pacman.d/mirrorlist:يخبر هذا pacman باستخدام ملف /etc/pacman.d/mirrorlist لاختيار أفضل مرآة لـ core و extra.**
- **[multilib]:هذا الخيار معطل افتراضيًا. إذا كنت بحاجة إلى دعم للتطبيقات 32-بت على نظام 64-بت، ستحتاج إلى إلغاء التعليق عليه (إزالة #).**
- **في سياق archiso: تتيح هذه الإعدادات لـ archiso الوصول إلى مستودعات Arch Linux الرسمية لتنزيل جميع الحزم التي قمت بتحديدها في packages.x86_64 و bootstrap_packages.x86_64.**

مستودعك المخصص (مثال(helwan :

[helwan]

SigLevel = Optional TrustedOnly

Server = https://helwan-linux.github.io/$repo/$arch

هذا القسم هو مثال حي على كيفية إضافة مستودعك المخصص.

- **[helwan]:اسم المستودع (يجب أن يكون فريدًا.)**
- **SigLevel = Optional TrustedOnly:هنا، أنت تخبر pacman أنه يفضل أن تكون الحزم موقعة(Optional) ، ولكنه سيثق بها إذا كانت تأتي من مستودع موثوق به (TrustedOnly) حتى لو لم تكن التوقيعات الصريحة موجودة. يمكن أن يكون هذا مفيدًا للمستودعات المخصصة حيث قد لا تكون كل حزمة موقعة باستمرار.**
- **Server = https://helwan-linux.github.io/$repo/$arch:هذا هو عنوان URL للمستودع.**
  - **سيتم استبدال repo$تلقائيًا باسم المستودع (أي.(helwan**
  - **سيتم استبدال arch$بالبنية المستهدفة (مثلاً.(x86_64**

لماذا يهم؟ يتيح لك هذا تضمين حزمك المخصصة، أو حتى الإصدارات المعدلة من حزمArch ، مباشرةً في صورة ISO الخاصة بك. ستحتاج إلى خادم ويب (أو حتى مستودع محلي) لاستضافة هذه الحزم.

المستودع المحلي المخصص (مثال(custom :

#[custom]

#SigLevel = Optional TrustAll

#Server = file:///home/custompkgs

هذا مثال آخر لمستودع مخصص، ولكنه يعتمد على نظام الملفات المحلي.

- **SigLevel = Optional TrustAll: هنا، يعني TrustAll أن pacman لن يتحقق من أي توقيعات على الإطلاق. لا يُنصح بهذا للمستودعات العامة بسبب المخاوف الأمنية ولكنه يمكن أن يكون مفيدًا للتجارب السريعة أو في البيئات المعزولة.**
- **Server = file:///home/custompkgs: يشير إلى مسار مجلد على نظام الملفات المحلي يحتوي على الحزم.**
- **في سياق archiso: إذا كنت تخزن حزمك المخصصة محليًا داخل ملفات البروفايل، يمكنك استخدام هذا النوع من الإعداد.**

مثال عملي توضيحي

لنفترض أنك تبني توزيعة "Helwan Linux" وتريد تضمين حزمة مخصصة اسمها helwan-tools غير متوفرة في مستودعات Arch الرسمية.

1. **أنشئ مستودعك المخصص:**
   - **أنشئ دليلًا لمستودعك، مثلاً ~/my_helwan_repo.**
   - **ضع ملف حزمتك (helwan-tools.pkg.tar.zst أو أي تنسيق آخر) داخل هذا الدليل.**
   - **أنشئ قاعدة بيانات المستودع باستخدام repo-add:**
2. **repo-add ~/my_helwan_repo/helwan.db.tar.gz ~/my_helwan_repo/helwan-tools-*.pkg.tar.zst**
   - **ارفع محتويات (~/my_helwan_repo بما في ذلك helwan.db.tar.gz و helwan-tools-*.pkg.tar.zst) إلى خادم ويب، أو استخدم file:// إذا كان محليًا.**
3. **عدّل pacman.conf: إذا قمت برفعه إلى https://helwan-linux.github.io/helwan/x86_64/، سيبدو قسم pacman.conf الخاص بك مثل المثال المقدم:**
4. **[helwan]**
5. **SigLevel = Optional TrustedOnly**
6. **Server = https://helwan-linux.github.io/$repo/$arch**
7. **عدّل packages.x86_64: أضف حزمتك المخصصة إلى القائمة:**
8. **... #حزم أخرى...**

9. helwan-tools

10. ... #حزم إضافية...

الآن، عندما يقوم archiso ببناء صورةISO ، سيعرف pacman من أين يجلب helwan-toolsوسيقوم بتثبيته في النظام الحي.

شرح إضافي(profiledef.shمن) pacman.conf

لقد قدمت أيضًا مقطعًا مثالًا من ملف profiledef.shيوضح استخدام pacman.confكملف خارجي:

Bash

"pacman.conf"=pacman_conf يشير إلى ملف pacman.conf الموجود في نفس دليل البروفايل

يشير هذا السطر في profiledef.shإلى أن archiso سيستخدم الملف المسمى pacman.conf، الموجود داخل نفس دليل البروفايل (مثل releng/أو archlive/~في مثالك)، كملف تكوين لـ .pacman هذا هو السلوك الافتراضي والموصى به، لأنه يحافظ على جميع إعدادات pacman مدمجة مع البروفايل الخاص بك.

ملخص الفصل

ملف pacman.confهو أداة قوية لتخصيص مصادر الحزم الخاصة بك والتحكم في كيفية جلب الحزم وتثبيتها. من خلال إدارته بعناية، يمكنك ضمان أن نظامك الحي لديه إمكانية الوصول إلى جميع الحزم التي يحتاجها، بما في ذلك الحزم المخصصة، بطريقة آمنة وفعالة. فهم هذه الإعدادات يمنحك تحكمًا كاملاً في "المستودعات" التي تعتمد عليها توزيعتك.