مقدمة: بناء نظام تشغيل مخصص

السلام عليكم. كان لديّ مشكلة في المساحة والوقت، مما منعني من العمل على كورس الفيزياء الذي كنا نعمل عليه. توقفتُ عن العمل لعدة أشهر. ستستمر هذه المشكلة على الأرجح حتى نهاية السنة، وربما بداية السنة القادمة إن شاء الله.

بدلاً من التوقف تماماً عن القناة، سأعمل على شيء لا يحتاج مساحة كبيرة، ويكون سهلاً في التحضير، بحيث أتمكن من إنجازه بسرعة وبجهد أقل. فكرة التركيز على الكمبيوتر أو أنظمة الكمبيوتر تبدو مناسبة في هذه المرحلة.

خلال العام الماضي، قمتُ بعمل فيديو أو اثنين عن أنظمة التشغيل والأدوات، وتلقيت بعض الأسئلة والطلبات حول مواضيع محددة، أو طلبات لمشاركة ملفاتي أو حساباتي على جيثب، إلخ. لم أرَ في ذلك فائدة كبيرة، لأن هذه المعلومات متوفرة بكثرة على الإنترنت. فإن بحثتَ عن "توزيع لينكس" أو "بايثون"، ستجد الكثير من المعلومات وقنوات كثيرة تغطي هذه المواضيع بالتفصيل. لذلك، لن أضيف شيئاً جديداً في هذا المجال.

نهج جديد: التركيز على النظام البيئي

ما سأفعله هو تناول النظام البيئي ككلّ. سأحدد احتياجاتي كمستخدم، وكيف أرغب في استخدام نظام التشغيل، وما هي الأدوات التي تعمل مع بعضها البعض، ولماذا اخترت هذه الأدوات تحديداً. سأتحدث حتى عن لوحة المفاتيح! سنركز على النظام البيئي بشكل شامل، وليس على أدوات منفصلة.

هذا الفيديو هو مقدمة نظرية فقط. هذه سلسلة صغيرة، ولن نتحدث عن أي شيء تقنيّ في هذه المقدمة، فقط لفهم كيفية سير الأمور. النموذج التخطيطي التالي يوضح شكل السلسلة:

```
[هاردوير] --> [نظام التشغيل] --> [البرامج] --> [المستخدم]
```

نتعامل مع نظام التشغيل، وليس مع الهاردوير مباشرة. نظام التشغيل يدير الهاردوير، ونحن نستخدم البرامج. سأتحدث لاحقاً عن نظام التشغيل بشكل مفصل.

أهمية النصوص

أهم ما يهمّني هو النصوص. لاحظ أن النص هو محور معظم استخدامات الكمبيوتر: كتابة رسائل البريد الإلكتروني، الدردشة، عناوين URL، المحتوى على الإنترنت، حتى واجهة المستخدم الرسومية في النهاية تُظهر نصاً. لذلك، أحتاج إلى الوصول المباشر إلى النصوص، والتعامل معها بكفاءة. لذا، يجب أن يتعامل النظام مع النصوص بشكل فعال.

متطلبات النظام

بما أننا نتعامل مع النصوص، نحتاج إلى لوحة مفاتيح جيدة. (سيكون هناك فيديو منفصل عن لوحة المفاتيح.) لكن، بما أن النظام يركز على النصوص، أحتاج إلى مواصفات معينة تتعلق بلوحة المفاتيح.

أنا لا أريد واجهة رسومية (GUI)؛ أريد واجهة نصية (TUI). أريد برامج سطر الأوامر، حتى لو كان لديّ تطبيقات رسومية، سأستخدمها كبرامج سطر أوامر. لستُ مضطراً لاستخدام لينكس أو TUI، لكني سأركز عليهما.

أحتاج إلى جودة عالية في واجهة النصوص، مع الحفاظ على إطار النص. لذلك، يجب أن يتعامل النظام مع واجهة المستخدم النصية بشكل جيد.

المتطلب الثاني: البرامج يجب أن تعمل مع فلسفة *Unix*. إذا لم تعمل أداة ما بشكل جيد مع *Unix*، فأنا لا أحتاجها.

نظام التشغيل

أحتاج إلى نظام تشغيل يدير الهاردوير، وفي نفس الوقت يُمكّنني من تثبيت وإدارة البرامج، ويسمح للبرامج بالتواصل معه. أحتاج إلى القدرة على تثبيت صور نظام التشغيل الأساسية (Kernel)، ثم توزيع لينكس (مثل Debian) فوقها، ثم تثبيت البرامج. لا أريد تثبيت صورة أساسية، ثم توزيع، ثم متصفح، وهكذا. أحتاج إلى نظام يتيح لي تثبيت صورة أساسية، مع مكتبة كبيرة من الحزم لأختار منها ما أحتاج. أريد أيضاً إصداراً "متدحرجاً" (Rolling Release)، للحصول على تحديثات البرامج والأمان بشكل مستمر.

لذلك، اخترت Arch Linux. إنه نظام مرن ورائع. لم أكن أرغب في استخدام أنظمة أخرى. أحببت فلسفته. و هذا ما دفعني لاختيار Arch Linux.

هل Arch Linux هو الخيار الوحيد؟ ليس بالضرورة، لكنه يناسب متطلباتي. يوجد العديد من التوزيعات الأخرى التي تقدم نفس المزايا، ولكن Arch Linux يتوافق مع فلسفتي.

نتحدث هنا عن أنظمة تشغيل شبيهة بـ *Unix*. Arch Linux هو توزيع *Linux*، وهو نظام *Unix-like*. يغطي Arch Linux العديد من الأجهزة، ويوفر مكتبة كبيرة من الحزم. لذلك، فهو خيار مناسب.

بناء النظام

أول شيء أحتاجه هو مدير نوافذ (Window Manager) و محطة طرفية (Terminal). بمجرد توفر هذين العنصرين، يمكنني تثبيت أي تطبيق آخر والبدء بالعمل.

سنقوم ببعض التعديلات (patching) على مدير النوافذ *dwm*، ولكن ليس كثيراً.

سنركز على إعداد لوحة المفاتيح بشكل جيد، لأن النظام يعتمد عليها بشكل كبير. سنتعرف على *dmenu* (قائمة) و *sxhkd* (لإدارة اختصارات لوحة المفاتيح). *sxhkd* هو برنامج يعمل في الخلفية للاستماع إلى لوحة المفاتيح، وعندما نضغط على مجموعة مفاتيح معينة، يحدث شيء محدد.

سنستخدم أيضاً *dwm* و *dmenu*، كلاهما من أدوات *suckless*. ليس بالضرورة استخدام *dmenu*، لكنني أستخدمه.

تحسينات إضافية

يمكننا استخدام محرر نصوص مثل *vim*، لكن ليس بالضرورة. يمكن استخدام *nano* أو أي محرر نصوص بسيط. سنتحدث عن *vim* بشكل سطحي.

سنناقش سلسلة من المواضيع: لون النص في المحطة، إعدادات الألوان، الطريقة التي يعمل بها نظام الألوان في المحطة، وكيفية دمج الألوان في التطبيقات. ستكون هذه المواضيع تقنية بعض الشيء، وقد تستغرق من 10 إلى 15 فيديو.

هذا الإطار هو الأساس. يمكنك تعديله حسب رغبتك. هذه هي رؤيتي لنظام التشغيل الخاص بي.

أتمنى أن يكون هذا مفيداً. سلام عليكم.

تثبيت Void Linux: شرح مفصل

مقدمة

{السلام عليكم}، في هذا الفيديو، سنقوم بعملية تثبيت نظام Void Linux. كما ذكرت في الفيديو السابق، ستكون مدة كل فيديو في هذه السلسلة حوالي 5-10 دقائق. لكن، اكتشفت أثناء تحضير هذا الفيديو أن عملية تثبيت Void Linux طويلة نوعًا ما، وسأركز في هذا الفيديو على عملية التثبيت فقط. سيستغرق الفيديو حوالي عشرين دقيقة تقريبًا. هذا يختلف عن توزيعات مثل Arch Linux، حيث تتضمن عملية التثبيت العديد من الخطوات الإضافية مثل تهيئة fstab وتكوين systemd وغيرها. لكن Void Linux تقدم فرصة جيدة لشرح بعض المفاهيم والمكونات المهمة في لينكس، ولن أضيع هذه الفرصة.

بدء عملية التثبيت

أولاً، سنذهب إلى الموقع الرسمي لـ Void Linux. ستجدون على الصفحة الرئيسية بعض المعلومات المهمة حول Void Linux وخصائصه المميزة. يمكنكم قراءتها بسرعة، لكنها ليست ضرورية لفهم عملية التثبيت. سننتقل مباشرةً إلى قسم التحميل.

سنجد أربعة منصات مختلفة يمكننا اختيارها للتثبيت. معظم أجهزة سطح المكتب الحديثة تعمل بنظام x86_64، وهذا هو النظام سنستخدمه.

ستلاحظون وجود خيارين لصورة التثبيت: صورة XFCE و صورة أساسية (Base Image). أفضل خيار هو اختيار الص الأساسية لأنها لا تتضمن أي بيئة سطح مكتب أو أدوات إضافية، مما يتيح لنا تثبيت البرامج التي نحتاجها فقط. تثبيت بيئة XFCE يعني تثبيت العديد من الحزم والبرامج التي قد لا نحتاجها.

اختيار صورة التثبيت: `live image` مقابل `root fs`

ستجدون نوعين من صور التثبيت: `live image` و `root fs`. سنقوم بتحميل كليهما.

- **`live image`**: هذه الصورة قابلة للتشغيل مباشرةً من USB أو قرص مضغوط، وتتضمن نظام التشغيل الأساسي، وبيئة سطح مكتب (إن تم اختيارها)، والأدوات اللازمة للتثبيت.
- **`root fs`**: هذا عبارة عن نظام جذر (/) مضغوط، وهو لا يحتوي على kernel أو برامج مثبتة مسبقاً. يُستخدم في حالات متقدمة، مثل عمليات التثبيت المشفرة أو على أنظمة ملفات خاصة مثل ZFS.

في هذا الفيديو، سنستخدم `live image` فقط، وسنقوم بحذف ملف `root fs`.

glibc مقابل musl

Void Linux يُقدم مكتبة C بديلة عن glibc وهي musl. glibc هي المكتبة الأكثر شيوعًا،
بينما musl تُعتبر أصغر وأكثر أمانًا وكفاءة
في استخدام الموارد. يُفضل استخدام musl في أنظمة مضغوطة أو عندما

يكون لديك عدد قليل من التطبيقات المثبتة. ولكن glibc
تدعم تطبيقات أكثر. في هذا الفيديو سنستخدم musl.

عملية التثبيت

بعد تحميل صورة musl (`live image`) ، سنقوم بنسخها إلى محرك أقراص USB
قابل للتمهيد. سأستخدم الأمر `dd` لهذا الغرض. **كن حذرًا جدًا عند استخدام الأمر dd، لأنه قد يؤدي إلى تلف محرك الأقراص إذا لم يتم استخدامه بشكل صحيح.**

```
dd if=path/to/image of=/dev/sdX bs=4M status=progress
```

(استبدل `/dev/sdX` باسم جهاز USB الخاص بك. تأكد من اختيار الجهاز الصحيح، لأن كتابة البيانات إلى جهاز خاطئ قد يؤدي إلى فقدان البيانات.)

بعد نسخ الصورة، سنقوم بتمهيد النظام من محرك أقراص USB. ستحتاج إلى تغيير إعدادات BIOS أو UEFI الخاصة بك لتحديد USB كجهاز التمهيد الأول.

اختيار إعدادات التثبيت

بعد التمهيد من USB، ستظهر لك قائمة الخيارات. سنختار اللغة، وتخطيط لوحة المفاتيح، وطريقة الاتصال بالشبكة (سأستخدم اتصالًا سلكيًا). سنختار أيضاً تثبيت النظام من الصورة المحملة على USB وليس من الإنترنت.

سنقوم بإنشاء حساب مستخدم جديد، مع تعيين كلمة مرور قوية. سنختار أيضًا مجموعة المستخدمين المطلوبة. سأترك فلابي درايف و سي دي روم فارغة لأنني لست بحاجة إليها.

تقسيم القرص الصلب

سنستخدم `cfdisk` لتقسيم القرص الصلب. سأنشئ ثلاثة أقسام:

1. قسم صغير (1 ميجابايت) من نوع BIOS Boot Partition لـ GPT.
2. قسم للبيانات (swap) بحجم 2 جيجابايت.
3. قسم رئيسي (root) لبقية مساحة القرص.

سيكون نظام الملفات للقسم الرئيسي ext4.

من المهم جدًا التأكد من أنك تقوم بتقسيم القرص الصحيح. خطأ واحد قد يؤدي إلى فقدان جميع بياناتك.

تثبيت النظام

بعد تقسيم القرص، ستبدأ عملية التثبيت. ستلاحظ أن عملية نسخ ملفات النظام (rootfs) ستستغرق بعض الوقت. بعد ذلك، سيقوم النظام بإنشاء initramfs وذلك لضمان عمل نظام التشغيل بشكل صحيح. بعد الانتهاء، سيتم إعادة تشغيل النظام.

عملية التمهيد

بعد إعادة التشغيل، ستبدأ عملية تمهيد النظام. سنقوم بشرح العملية بشيء من التفصيل:

أولاً، يقوم bootloader (GRUB) بالبحث عن أنظمة التشغيل المثبتة. ثم، يقوم بتحميل kernel إلى الذاكرة. بعد ذلك، يقوم ا
بتحميل initramfs، والذي يحتوي على drivers و scripts ضرورية لبدء تشغيل النظام. ثم يقوم initramfs بتحميل نظام ا
بتشغيل كافة الخدمات اللازمة.

initramfs و أهميته

`initramfs` يحتوي على المعلومات والبيانات الضرورية لتمهيد النظام. من أهميته:

- **تشفير القرص:** إذا كان لديك قرص مشفر، فإن `initramfs` يحتوي على المفتاح اللازم لفك تشفير القرص قبل تحميل نظام الملفات.
- **أنظمة ملفات غير مدعومة:** إذا كان نظام الملفات الخاص بك غير مدعوم من قبل kernel بشكل افتراضي، فإن `initramfs` يحتوي على drivers اللازمة لقراءته.

BIOS مقابل UEFI

يجب معرفة الفرق بين BIOS و UEFI قبل البدء بعملية التثبيت. UEFI هو معيار أحدث وأكثر أمانًا من BIOS، ويستخدم في أجهزة الكمبيوتر الحديثة. قد تحتاج إلى ضبط إعدادات UEFI لتتمكن من تمهيد النظام من USB.

MBR مقابل GPT

MBR و GPT هما طريقتان لجدولة الأقسام على القرص
الصلب. GPT
هو معيار أحدث وأكثر مرونة، ويدعم أقراصًا أكبر من 2 تيرابايت. يُنصح
باستخدام GPT مع UEFI.

خاتمة

هذا شرح مفصل لعملية تثبيت Void Linux. آمل أن يكون هذا الفيديو مفيدًا لكم. {سلام عليكم ورحمة الله وبركاته}

تثبيت Void Linux و DWM و ST: دليل خطوة بخطوة

مقدمة

السلام عليكم. في هذا الفيديو، سنستكمل بناء نظامنا، وسنقوم بتثبيت بيئة العمل الرسومية DWM ومدير النوافذ ST. سأشرح الخطوات بالتفصيل، كما لو كنت تقوم بالتثبيت لأول مرة. ستأخذ عملية التثبيت نفسها معظم وقت الفيديو.

تحديث النظام

قبل البدء، من المناسب تحديث النظام. في الأسبوع الماضي، قمنا بتحديث بعض الحزم، لذا النظام مُحدث بالفعل. لا يوجد الكثير من الحزم التي تحتاج إلى التحديث.

تثبيت البرامج

عند تثبيت البرامج، فإن المصدر الرئيسي الأول الذي يجب التفكير فيه هو مستودعات التوزيع. للتثبيت، سنستخدم `pacman`. سأستخدم الأمر `pacman -Syu` لتحديث النظام.

تثبيت DWM

سنتناول الآن تثبيت DWM. تتميز DWM بعدم وجود ملف تكوين. التغييرات تُجرى مباشرة في الكود المصدري. هذا يجعل الكود أصغر وأسرع وأكثر كفاءة في الذاكرة. لكن الجانب السلبي هو أن الكود يصبح أكبر، لأنه يجب تغطية جميع الاحتمالات.

في حالة DWM، التكوين كله يتم من خلال ملفات المصدر. هذا يجعل العملية أكثر تعقيدًا، ولكنه يضمن سرعة وأداء أفضل.

سنستخدم `git` لتحميل الكود المصدري لـ DWM. سأقوم بتحميله إلى مجلد `/home/user/src/dwm/`.

بعد تحميل الكود، نحتاج إلى إنشاء ملف `Makefile`. هذا الملف يحتوي على الأوامر اللازمة لتجميع DWM. سأستخدم محرر `vim` لتحرير الملف.

بعد تجميع DWM، سنقوم بتثبيته باستخدام الأمر `make install clean`. قد تواجه بعض الأخطاء أثناء التجميع، مثل خطأ `xlib.h` أو `xft.h`. هذه الملفات موجودة في حزم تطوير X11، لذلك علينا تثبيتها أولاً. سنحتاج أيضًا إلى تثبيت حزم تطوير `xorg`.

تثبيت Xorg

لإطلاق DWM، نحتاج إلى تثبيت خادم العرض Xorg. Xorg هو خادم العرض القياسي في لينكس. سأقوم بتثبيت `xorg-server`. عملية بدء Xorg معقدة نوعًا ما، لذا سنستخدم برنامج `startx` الذي يعمل كلفافة (wrapper) لإدارة بدء وإيقاف Xorg. سوف نحتاج أيضاً إلى تثبيت `xinit`.

بعد تثبيت `xorg` و `xinit`، سنحاول تشغيل `startx`. قد تواجه رسالة خطأ "can't open display". هذا يعني أن DWM لا يستطيع الاتصال بخادم العرض.

حل مشكلة "can't open display"

لحل هذه المشكلة، علينا التأكد من تشغيل خادم العرض Xorg بشكل صحيح. بعد تشغيل `startx`, قد تجد أن DWM لا يزال لا يعمل بشكل صحيح. قد يكون ذلك بسبب نقص بعض السائقين. سنقوم بتثبيت سائقين الإدخال باستخدام الأمر `pacman -S xorg-xinput`. بعد ذلك، أعد تشغيل `startx`.

تثبيت ST

بعد تثبيت DWM و Xorg، سنقوم بتثبيت ST (Suckless Terminal). سنستخدم `git` لتحميل ST وتجميعه وتثبيته باستخدام `make install clean`.

ضبط DWM

بعد تثبيت جميع البرامج، سنقوم بضبط DWM. سنقوم بتعديل ملف التكوين الخاص به، وتغيير بعض الإعدادات، مثل الخطوط. سنحتاج إلى تثبيت خطوط Mono Space مثل `xorg-fonts`.

الخاتمة

وهنا نكون قد انتهينا من تثبيت Void Linux و DWM و ST. لديك الآن نظام تشغيل كامل الوظائف، ويمكنك تثبيت أي برنامج ترغب به. سنستكمل في الفيديو القادم بتفاصيل أكثر عن DWM وخصائصه. السلام عليكم.

الكيبورد: رحلة في عالم المفاتيح

السلام عليكم! زي ما هو واضح من الفيديو، إن شاء الله هيكون كلامنا عن الكيبورد. الكيبورد مش هو محور حديثنا الأساسي، أنا ما كنتش محتاجة أعمل القصة دي كلها، كنت محتاجة بس أتكلم عن نوع الكيبورد اللي بستخدمه، ووصف المفاتيح. كنت ممكن أصور الكيبورد على الكمبيوتر وأسجل الفيديو وأخلص، بس بصراحة في نقطتين مهمتين محشورين في دماغي من زمان بخصوص الكيبورد، فقلت نستغل الفرصة ونتكلم عنه بشكل مُناسب.

معلش، التربيزة دي مش مكاني المعتاد للتسجيل، فالصوت والإضاءة مش هيكونوا مثاليين، استحملوني شوية. التربيزة دي مناسبة لعرض الجهاز، لكنها نفس التربيزة اللي بعمل عليها قهوتي! وبالمناسبة، دي آلة إسبريسو يدوية، ودي مطحنة حبوب البن (لازم تطحن البن قبل ما تعمل قهوة!). مش هنعمل فيديو عن القهوة، بس بالمرة أوضّح نقطة مهمة لهواة القهوة.

نرجع لكلامنا عن الكيبورد. هنتكلم عن أمور كتيرة، ومدح بعض الحاجات، وانتقاد أخرى. رأيي قوي جدًا في بعض النقاط، وربما مش كل الناس هتتفق معايا، ومفيش مشكلة. الكلام ده ما يعنيش إنك تروح تشتري كيبورد جديدة، لو عندك كيبورد شغالة، حتى لو بسيطة، استخدمها. هنتكلم عن بعض النقاط اللي ممكن تنتقدها، ونتعلم مع بعض. بصفتنا قناة علمية، أغلب المشاهدين بيكونوا متخصصين، فلا داعي لشراء كيبورد جديدة إلا إذا كنت محتاجها فعلاً. خلي الكلام ده في اعتبارك قبل ما تختار كيبورد.

أهم مكونات الكيبورد

في نوعين من المستخدمين: نوع بيستخدم كيبورد، ونوع بيستخدم... (شيء آخر). وظيفتي في الفيديو ده أوضح المشكلة والحلول. أول حاجة مهمة في الكيبورد هي *السويتش*. لما بتدوس على زر، في حاجة تحت بتعمل اتصال بدائرة كهربائية، وده بيشغل الكيبورد. السويتش هو أهم جزء. في البداية، كانت الكيبورد ميكانيكية، كان فيه جزء بيمس دائرة كهربائية، وبعدين بيرجع مكانه. بعد فترة، الشركات عملت كيبورد أرخص باستخدام *رابر دوم*. الرابر دوم عبارة عن قباب مطاطية، لما بتدوس بتضغط على حاجة بتقفل الدائرة.

الكيبورد بالرابر دوم ده *سيء* جدًا، بعد فترة بتبقى جامدة. المفروض الكيبورد تكون ميكانيكية. أعرف إن الكيبورد الميكانيكية أغلى شوية، ودي نقطة مهمة في كلامي السابق عن عدم شراء كيبورد جديدة إلا إذا كنت محتاجها. لكن الكيبورد الميكانيكية بتعيش معاك طول عمرك، أما كيبورد الرابر دوم جودتها بتقل مع الوقت. بتبقى خشنّة وجامدة، وممكن تتكسر.

الكيبورد الميكانيكية الحقيقية بتشمل: الباكلت لايتينج، السبرينج، السويتشات... إلخ. الباكلت لايتينج نادر، وما عنديش خبرة كبيرة فيه. السويتشات موضوعها طويل، فيه شركات كتيرة بتعمل سويتشات، أشهرهم *تشيري*. تشيري شركة ألمانية قديمة جدًا، عارفين شغلهم كويس. فيه شركات تانية زى *كات*، و*أنجيترون*. أنا بستخدم تشيري بشكل أساسي. تشيري ميكس فيه أنواع أساسية: بلو، براون، وريد.

(هنا يضع المُحاضر بعض الصور للكيوردات)

دي بلو سويتش، ودي براون، ودي ريد. الصوت بتاع الريد أقل من البراون، وأقل من البلو. البلو صوتها عالي جدًا. موضوع الكيبورد *شخصي جدًا*، كل واحد وله طريقته في الكتابة. تجربتك مع الكيبورد هتختلف من شخص لشخص. الكتابة على البلو رائعة مع النصوص العادية، لكنها سيئة مع الرموز المتكررة. في الحروف المتكررة، بتحس بثقل في الضغط، وبتعمل أخطاء كتيرة. جودة الكتابة مش عالية، لكنها أفضل مع الرموز الخاصة. البرنامجين بيستخدموا براون عادة، مش بس عشان الصوت، لكن عشان هي أفضل مع الرموز الخاصة. أنا مبسوط جدًا بالبراون، لكن الصوت العالي ممكن يزعج. لو هاشتري كيبورد جديدة، هاشتري براون أو ريد، لكن مش بلو.

أنواع الكيز

الكيز نفسها فيه نوعين: *كي كابيتي* و*بي بي تي*. الكابيتي أرخص وأشهر، أكبر حجمًا، لكن جودته أفضل من بي بي تي. بي بي تي تصنيعه أصعب، وأكثر هشاشة. ده رأي الناس، وكل واحد وله طريقته. الكي كابيتي لمعانه مش ثابت. أنا بفضل كي كابيتي. الكيكاب فيه نسيج بيعمل احتكاك مع الصابع، أنا بفضل الكيكاب اللي نسيجه ناعم.

ال- ABS، كلنا عارفين مشكلته، الحروف بتتمسح، لأن البلاستيك بيمتص الزيوت من صوابعك. الحروف المطبوعة على الكيكاب بتتمسح، لكن الحروف المنقوشة من جوا بتفضل موجودة. أنا بفضل الكيبورد من غير طباعة. فيه كيبورد بتعمل كده، لكن عندها عيوب تانية.

حجم الكيبورد

دي كيبورد عادية، ودي كيبورد Tenkeyless (*TKL*)، من غير نيمباد. ال- TKL
بتاخد مساحة أقل، ومريحة أكتر، النمباد مش ضروري لمعظم الناس. ممكن تحتاج النيمباد لو كنت محاسب أو مهندس شبكات، لكن حتى في الحالتين دي مش هتحتاجها دايما. الصف اللي فوق ده (الفانكشن كيز) ملوش لازمة بالنسبة لي. فيه كيبورد 60%، من غير فانكشن كيز،
و
arrows. مريحة مع أغلب الاستخدامات، لكن ممكن تحتاج ال- arrows
في بعض الأحيان. كيبورد 65% بتجمع بين المفاتيح الأساسية وبعض مفاتيح
الأسهم.

بروفايل الكيز

الكيبورد فيها كي بروفايل، اللي هو شكل المفاتيح وزاوية انحنائها. فيه أنواع كتيرة، الأشهر أربعة أو خمسة. أنا بفضل البروفايل الأسرع شوية.

كيبورد لابتوب

فيه ناس بتشتري كيبورد زي اللي في اللابتوب، ده غلط جدًا! مفاتيح اللابتوب مافيهاش مسافة كافية بينها. معظم اللابتوبات دلوقتي كده، حتى الثينكباد. صعب تحط سويتشات ميكانيكية في اللابتوب.

سوفتوير الكيبورد

فيه كيبورد بتيجي معها سوفتوير، وده ممكن يكون في الفيرم وير أو منفصل. أنا بفضل اللي في الفيرم وير. سوفتوير الكيبورد غالبًا *كلوز سورس*، ومش بيشتغل على لينكس. الكيبورد الاحترافية بتستخدم سوفتوير مفتوح المصدر.

اللي أوت (ISO vs ANSI)

هنتكلم عن اللي أوت، فيه نوعين: ISO و ANSI. ال- ISO معمول للأوروبيين، ال- ANSI أفضل. ال- ISO فيه زرار إنتر كبير، وال- ANSI زرار إنتر صغير. في اللغات الأوروبية، نفس الحرف ممكن يكون له أكتر من شكل. ال- ISO بتحاول تتكيف مع ده. في الوطن العربي، ما نحتاجش الكلام ده، نستخدم ANSI. أنا عمري ما اشتريت كيبورد ISO.

الـANSI فيه زرار Shift متماثل الحجم والمسافة على الجانبين، ده مهم جدًا. فيه نقاش بين الأوروبيين أنفسهم عن كفاءة ISO و ANSI. بعض المبرمجين بيقولوا إن ANSI أكتر كفاءة.

الموديفاير كيز

الموديفاير كيز مهمة جدًا، خاصة في أنظمة تشغيل مثل لينكس. مهم جدًا إن الموديفاير كيز تكون موجودة على الجانبين، عشان تبقى إيديك على الهوم رو. أنا بستخدم كنترول، شيفت، وسوبر. الكابسلوك ملوش لازمة، أفضل أحطه مكان الإسكيب.

سبلت كيبورد

الكلام ده كله ينطبق على الكيبورد العادية، مش السبلت كيبورد. السبلت كيبورد فيه مفاتيح منفصلة، ده بيخلي إيديك في وضع أفضل. ميزة السبلت كيبورد هي *الثم كلاستر*. الثم كلاستر بيخليكي تدوس على أكتر من زر بنفس الوقت، وبالتالي بتنتفي الحاجة للموديفاير كيز على الجانبين. ده الميزة الوحيدة المهمة في السبلت كيبورد. مشكلتها أنها بتاخد مساحة أكبر.

تجارب شخصية

(هنا يضع المُحاضر صورًا لأجهزة الكمبيوتر التي يملكها)

دي كيبورد قديمة، ودي كيبورد حديثة، ودي لابتوب قديم، ودي لابتوب جديد. الكيبورد دي اشتريتها زمان، كانت جيدة وقتها، لكن مع الوقت اكتشفت أنها مش مناسبة لي. الكيبورد دي من عصر الجاهلية! لكن ده بيعلمني إن أفكارنا وتفكيرنا بيتغيّر مع الوقت. اللي بنوصله هو نتيجة تجاربنا. البيئة اللي حواليك بتأثر عليك. حاول تحيط نفسك بناس بتحسن من حياتك.

شكرًا لكم، والسلام عليكم.

فهم آلية تصحيح الأخطاء (Patching) في WMOST

مقدمة

السلام عليكم، كان من المفترض أن نتحدث في هذا الفيديو عن تصحيح الأخطاء (Patching) باستخدام WMOST، لكن سنؤجل هذا الموضوع للمرة القادمة. سنركز اليوم على استخدام أدوات مثل DWM. في الفيديو السابق، قمنا بتحميل DWM من Get أو من مصدر آخر، وقمنا بتجميعه. لدينا الآن البرنامج، وهو الرمز الأصلي من سكلس بدون أي ميزات إضافية.

إضافة التصحيحات (Patches)

إذا أردت إضافة تصحيح، يوجد قسم خاص بذلك في كل ملف برمجي في DWM. يحتوي هذا القسم على تصحيحات قام بها مستخدمون آخرون، ويمكنك استخدامها. عند فتح أي تصحيح، ستجد صورة توضح وظيفته، ووصفًا تفصيليًا، وأهم جزء وهو ملف `.diff`.

يحتوي ملف `.diff` على بناء جملة قد يبدو غريباً إن لم تكن معتادًا على ملفات `.diff` وعمليات التصحيح. لكن الأمر بسيط؛ فهو يحتوي على التعديلات التي تُطبق على الكود، سواء بإضافة ميزات جديدة أو إزالتها. سنشرح آلية تطبيق التصحيحات في الفيديو القادم.

تطبيق التصحيحات: أمثلة عملية

دعونا نوضح ذلك من خلال أمثلة. لدي هنا مجلد يحتوي على ملفات `.diff`. سأستخدم ثلاثة مجلدات، كل منها يمثل سيناريو مختلفًا، لشرح العملية بشكل مفصل.

سيناريو 1: ملف واحد

لنبدأ بـ "Case 1". لدينا ملف `A.txt`. لنفترض أن شخصًا ما قام بنسخ هذا الملف وعمل عليه بعض التعديلات، وحفظ النسخة المعدلة باسم `B.txt`. الآن، نريد تطبيق هذه التعديلات على الملف الأصلي `A.txt`. الأمر بسيط للغاية: نقوم بكتابة `A.txt` فوق `B.txt`، وبذلك يتم تطبيق التغييرات.

سيناريو 2: تعديلات متعددة

لننتقل إلى "Case 2". لنفترض أن شخصين قام كل منهما بنسخ الملف `A.txt`، وقام كل منهما بإجراء تعديلات مختلفة. شخص قام بتعديل أول ثلاثة أسطر، وآخر قام بتعديل ثلاثة أسطر أخرى في قسم مختلف. كيف نطبق هذين التعديلين على الملف الأصلي؟

في هذه الحالة، لا يمكننا فقط كتابة ملف `.diff` على ملف `.diff` آخر، لأننا سنفقد بعض التغييرات. سنحتاج هنا إلى فهم كيفية توليد ملف `.diff` وبناء جملته. سنتعرف على هذا الأمر بشكل مفصل، وسنرى كيف نستخدم الأمر `diff` لإنشاء ملف `.diff`، وكيف نستخدم الأمر `patch` لتطبيقه.

توليد ملف `.diff` و تطبيقه

دعونا نستخدم الأمر `diff -u A.txt copy1.txt` لإنشاء ملف `.diff` من التغييرات الموجودة في `copy1.txt` مقارنة بـ `A.txt`. خيار `-u` يضمن الحصول على بناء جملة مفصل. سيكون الناتج عبارة عن سطور تبدأ بـ `-` (إزالة) و `+` (إضافة). هذه الرموز ليست عشوائية، بل تشير إلى التغييرات التي تم إجراؤها في ملف `.diff`. كل سطر في الملف الأصلي مرتبط بعلامة ناقص (-)، وكل سطر مضاف مرتبط بعلامة زائد (+). يستخدم `patch` هذه المعلومات لتطبيق التغييرات بدقة. يحتوي ملف `.diff` أيضاً على سياق (Context) قبل التعديل وبعده للتأكد من تطبيق التصحيح في المكان الصحيح.

سيناريو متعدد التعديلات (متابعة)

في حالة "Case 2"، يحتاج `patch` إلى سياق (Context) للتأكد من تطبيق التصحيحات في المواقع الصحيحة. إذا كان لديك تعديلات متعددة من مصادر مختلفة، يجب تطبيقها بالتسلسل الصحيح لتجنب فقدان أي تغييرات.

في الختام، فإن فهم آلية توليد ملفات `.diff` و استخدام الأمر `patch` أمرٌ بالغ الأهمية لتطبيق التصحيحات بشكل صحيح وفعال. نتمنى أن يكون هذا الشرح مفيداً، وسنواصل شرح هذا الموضوع بشكل أكثر تفصيلاً في الفيديوهات القادمة.

باتشينج DWM: دليل شامل

سلام عليكم. في هذا الفيديو، سنعمل على باتشينج DWM. قبل الخوض في التفاصيل، أود توضيح بعض النقاط المهمة. هناك جملة شائعة على اليوتيوب تقول أن DWM برنامج جيد، ولكنه غير قابل للاستخدام بدون باتشينج. هذه الجملة غير صحيحة تماماً، كما سنرى. فـ DWM يملك وظائف أساسية محدودة جداً، وغير مرتبطة بمعظم وظائفه الإضافية. معظم الباتشات لا تضيف وظائف جديدة، بل تعدّل الوظائف الحالية فقط. هناك باتشان أو اثنتان تضيفان وظائف مهمة، ولكنها خاصة بحالات استخدام محددة.

باختصار، DWM برنامج بسيط يفتح نافذة للبرامج، وظائفه الأساسية بسيطة ولا تحتاج إلى الكثير من الباتشات. لكن هذا لا يعني عدم استخدام الباتشات. الباتشات التي تستخدمها يجب أن تكون مرتبطة بطريقة استخدامك للنظام، وأن تكون ضرورية لك. لا تستخدم باتشات تستخدمها مرة كل ستة أشهر، أو باتشات لتسهيل استخدام باتشات أخرى.

دعونا الآن نبدأ باتشينج DWM، باستخدام نفس النظام الذي استخدمناه من بداية السلسلة: XORG، و DWM، و startx. سنفتح startx، والذي سيفتح XORG و DWM بشكل افتراضي، بدون أي باتشات. هذا هو DWM الأصلي. في هذا الفيديو، لن نقوم بأي تكوين، وسنركز فقط على الباتشات. التكوين سيكون في فيديو لاحق إن شاء الله. طريقة التخصيص في DWM هي إضافة الباتشات ثم تطبيق التكوين.

لفتح ترمينال، استخدم اختصار لوحة المفاتيح "Alt + Shift". بعض المشاهدين سألوا عن كيفية تغيير حجم النص. يمكنك استخدام "Ctrl + Shift + Page Up" لتكبير النص، و
"Ctrl + Shift + Page Down" لتصغيره، و "Ctrl + Shift + Home" لإعادة حجم النص إلى حجمه الأصلي. سنتحدث عن تغيير الخطوط وحجمها في فيديو لاحق إن شاء الله.

البدء بالباتشينج

لنبدأ بفتح Firefox. (تكوين Firefox سيكون في فيديو لاحق). سنذهب إلى موقع Sacles tools، ونبدأ العمل على الباتشات. سأتحدث عن باتش محددة هنا، وهي باتش "Adjacent Tag". أنا شخصياً لا أحتاجها، ولن أستخدمها في هذا الفيديو، لكني أردت ذكرها لأسباب خاصة. أود أن أشكر أيوب خاطر على هذه الباتش، فهو من أعضاء القناة، وساهم كثيراً في تطويرها. الشيء الرائع هو أن لدينا شخصاً في القناة يقوم بعمل باتشات بلغة Dyalect، وهو أمر نادر الحدوث. أيوب من أكثر ودّاً ومساعدةً في القناة، وهو مثال رائع على التعاون بين أعضاء المجتمع. جزاه الله خيراً.

قسم التخصيص في DWM

عند فتح DWM، ستجد قسم "التخصيص" الذي يحتوي على عدة خيارات، مثل الوظائف المخصصة، والخطوط، والقواعد، إلخ. أقترح قراءة هذا القسم جيداً. سنتحدث عن بعض النقاط المهمة هنا، لكن بشكل عام، هذا القسم مهم جداً.

فهم هيكل الملفات في Sacles tools

قبل التحدث عن الباتشات التي سنستخدمها، هناك بعض النقاط المهمة حول هيكل الملفات في Sacles tools. هذا الجزء مهم جداً. أستخدم أسلوباً معيناً في كتابة الكود، حيث أضع كل الكود المصدر في مجلد اسمه "src". ثم أقوم بإنشاء ملفات `.h` و `.dev` و `.make`. يمكنك تجاهل ملف `.make`، فهو خاص بالتكوين والتخصيص. أنا شخصياً لا أغير أي شيء فيه. الجزء المهم هو فهم ملفات `.h` و `.dev` و وظائفهما والفرق بينهما. في معظم الأحيان، ستجدان متطابقتين.

سنبدأ بتوضيح ماهية هذين الملفين، ثم سنتحدث عن كيفية استخدامهما مع الباتشات. لكي تتمكن من استخدام الباتشات وتخصيص DWM بشكل فعال، يجب عليك فهم كيفية عمل هذين الملفين. هناك عدة طرق لاستخدامهما، وسأشرح بعضها.

DWM وملحقاته لا تأتي مع ملف `.dev` بشكل افتراضي. عند تنزيل الكود المصدر، لن تجد هذا الملف. لكن ملف `.h` موجود ضمن الكود المصدر. إذا قمت بتنفيذ الأمر `grep config.h`، ستجد هذا الملف في ملفات `Makefile`، و `README`، و `help`، و `dwm.s` (وهو ملف `include`). أما ملف `.dev`، فهو موجود فقط في ملف `Makefile`. يستخدم ملف `.h` في عملية الترجمة لأنه مُدرج في الكود المصدر. أما ملف `.dev`، فهو لا يُستخدم بشكل مباشر في الكود.

ملف `.dev` يُنشأ من خلال ملف `Makefile` من ملف `.h` الذي يأتي كقالب من الكود المصدر. إذا كان ملف `.dev` موجوداً بالفعل، فلن يُنشأ ملف جديد. هذا يعني أن ملف `.dev` الأصلي لا يأتي مع الكود المصدر، بل يُنشأ خلال عملية الترجمة من ملف `.h` إذا لم يكن موجوداً. بمعنى آخر، ملف `.dev` هو الملف الذي يحتاجه المُترجم، وهو موجود في الكود عن طريق `include`. الهدف من هذا هو السماح لك بتعديل الإعدادات في ملف `.dev` بدون تغيير الكود الأصلي.

الباتش الأول: إزالة شريط العنوان (No Title)

أول باتش سنعمل عليها هي إزالة شريط العنوان. أنا لا أحب شريط العنوان في DWM، وأفضل إزالته. هناك باتش اسمها "no title" تقوم بذلك. هذه الباتش تزيل شريط العنوان، وتترك فقط شريط الحالة. هناك باتشات أخرى تقوم بنفس المهمة، ولكن مع بعض التعديلات الإضافية.

أنا أستخدم أيضاً باتش أخرى تُقسم شريط الحالة إلى قسمين: قسم على اليسار، وقسم على اليمين. هذا يسمح لي بعرض معلومات إضافية، مثل تسجيل الشاشة. اسم هذه الباتش "split status". هذه الباتش تحتاج إلى تعديل في ملف `Makefile`.

سنقوم بنسخ رابط الباتش، ونقوم بتحميلها باستخدام `wget`. بعد ذلك، سنقوم بتطبيقها. سأوضح طريقة وضع ملفات الباتش، أنا شخصياً أفضل وضعها في مجلد منفصل خارج مجلد الكود المصدر. سأقوم بتحميل `split status` ثم تطبيقها.

الباتش الثاني: إزالة الإطار (No Border)

الباتش الثانية هي إزالة الإطار من النوافذ. أنا لا أحتاج الإطار إلا عندما يكون لدي أكثر من نافذة مفتوحة. هناك باتش اسمها "no border" تقوم بذلك. هذه الباتش بسيطة جداً، ولا تقوم بأي تعديلات في وظائف DWM. سوف نقوم بتحميل هذه الباتش وتطبيقها. لاحظوا أنه لا يوجد تعديل في ملف `.h`، لذلك لا حاجة لإزالته قبل الترجمة.

الباتش الثالثة: Push (تحريك النوافذ في المكدس)

الباتش الثالثة هي "push"، والتي تسمح بتحريك النوافذ في المكدس. أنا لا أستخدمها كثيراً، ولكنها مفيدة في بعض الأحيان. هذه الباتش تقوم بتعديل في ملف `.h` ولذلك يجب عليك اضافتها يدوياً.

الباتش الرابعة: Scratchpad (نافذة مؤقتة)

أخيراً، سأتحدث عن Scratchpad. هذه الوظيفة تسمح بفتح نافذة مؤقتة تختفي عندما لا تكون بحاجة إليها. هناك العديد من الباتشات التي تقدم هذه الوظيفة، وكل منها لها طريقة عمل مختلفة. أنا أستخدم باتش "centered scratchpad" والتي تضع النافذة المؤقتة في المنتصف.

هذه هي الباتشات التي أستخدمها في DWM. الباقي هو فقط تكوين ملفات التكوين الخاصة بالاختصارات، والخطوط، وحجم الخطوط، إلخ. أنا لا أقوم بأكثر من ذلك. هناك باتش واحدة فقط أستخدمها في ST (terminal)، وهي "scrollback". أنا أستخدم Tmux، والذي يقدم وظيفة scrollback أفضل.

هذا كل شيء. آمل أن يكون هذا الدليل مفيداً. سلام عليكم.

الكي مودي فايرز: اختيارك الأمثل

سلام عليكم. في هذا الفيديو، سنتحدث عن موضوع بسيط ولكنه مهم جدًا: الكي مودي فايرز. تأثير الكي مودي فايرز يكون كبيرًا ومباشرًا على أسلوبك في استخدام لوحة المفاتيح. بما أن النظام الذي نبني عليه يعتمد بشكل كبير على لوحة المفاتيح، فهي تعتبر من العناصر المركزية، وبالتالي فإن هذا الموضوع مهم جدًا، خاصةً في هذا الفيديو الذي نتحدث فيه عن تهيئة (كونفجرشة) دي دي وموستي. لقد قمنا بتخصيص دي دي وموستي، ولكن لا تزال هناك بعض التهيئات التي نحتاج إلى إجرائها، خاصةً فيما يتعلق بتعيين اختصارات لوحة المفاتيح (كي بايندنز). ستكون الكي مودي فايرز من الأمور المهمة في هذا الجزء.

ما هي الكي مودي فايرز؟

معظم الكلام الذي سأقوله موجود في الرسم التخطيطي (الدايجرام) الموجود في الفيديو. سأقدم بعض الأمثلة البسيطة، لكني سأحاول أن يكون الفيديو قصيرًا. المبدأ الأساسي هو: ما هي الكي مودي فايرز؟ الكي مودي فايرز الموجودة في معظم لوحات المفاتيح هي: Alt و Windows و Shift.

كما ذكرت سابقًا، من أهم خصائص لوحة المفاتيح الجيدة أن تكون الكي مودي فايرز موجودة على كلا الجانبين. لماذا؟ لأنني عندما أضع يدي على الصف الرئيسي (الهوم رو) أريد العمل باستخدام كلا الجانبين، وفي نفس الوقت أريد أن أكون قادرًا على الوصول إليها إذا كنت أستخدم يدًا واحدة. على سبيل المثال، إذا كنت أحتاج لرفع الصوت، أريد أن تكون الكي مودي فايرز على كلا الجانبين حتى أتمكن من التعامل مع أي جانب من لوحة المفاتيح. هذه نقطة مهمة جدًا. طبعًا، هذا الكلام يختلف قليلاً إذا كنت تستخدم لوحة مفاتيح منقسمة (سبلات كي بورد)، لكني هنا أتحدث عن لوحات المفاتيح العادية.

تسمى هذه الأزرار "كي مودي فايرز" لأنها تعدّل (موديفاي) وظيفة أزرار أخرى. على سبيل المثال، حرف "h" صغير، وبالضغط على Shift + h يصبح حرف "H" كبير. وظيفتها ليست العمل بمفردها (الضغط على Ctrl وحده لا يفعل شيئًا)، بل تعديل مفتاح آخر لإجراء وظيفة جديدة.

اختيار الكي مودي فايرز

الطريقة التي سنتبعها في هذا الفيديو لاختيار الكي مودي فايرز هي: سنبدأ بإزالة الخيارات غير المناسبة، ثم نختار من الخيارات المتبقية.

- **Shift:** من الواضح جدًا أنه اختيار غير مناسب، لأنه يستخدم مع كل شيء تقريبًا على لوحة المفاتيح، بما في ذلك الكي مودي فايرز الأخرى. استخدام Shift ككي مودي فاير هو اختيار سيء تقريبًا. هذا الكلام ينطبق على أي مدير نوافذ (ويندومانجر) وأي محطة (ترمينا).

- **Ctrl و Alt و Windows:** لاحظ في الرسم التخطيطي أن نافذة DWM (مدير نوافذ سطح المكتب) محاطة بإطار، وهذا يعني أنها خاصة بالنوافذ المفتوحة. عندما تكون لديك نافذة مفتوحة، فإنها مفتوحة بسبب تشغيل برنامج ما. لذلك، عندما تتفاعل مع النافذة عبر DWM وكي بايندنز، يجب أن تضع في اعتبارك أنك لا تتفاعل مع النافذة فقط، بل مع البرنامج الذي فتحها. لذلك، من المهم تجنب تعارضات (كونفلكت) في تعيين اختصارات لوحة المفاتيح. معظم البرامج تستخدم Ctrl و Alt وحتى Shift، لذلك ليس من المنطقي استخدام هذه المفاتيح ككي مودي فايرز للتفاعل مع DWM، وإلا فستكون هناك تعارضات. هذا ليس أمرًا مؤكدًا بنسبة 100%، بل يعتمد على نوع البرنامج. لكنني أريد تجنب هذه المشكلة.

لاحظ أنني وضعت أقواس حول بعض الكلمات. سأشرح هذا لاحقاً.

حتى في المحطة (الترمينا)، تستخدم البرامج مثل Bash مفتاح Ctrl بكثرة. لذلك، Ctrl مهم أيضًا لتطبيقات المحطة.

الخيار الأمثل: Windows Key

بناءً على ما سبق، فإن الخيار الوحيد المتبقي هو Windows Key (أو Super Key). هذا منطقي لأن معظم البرامج لا تستخدم هذا المفتاح كثيرًا، لذا من غير المرجح وجود تعارضات.

هذا ليس شرطًا، فهذا مجرد تفضيل شخصي. أعطيك قاعدة انطلاق، وإذا كان لديك نظام يعمل بشكل جيد، فابق عليه.

التفاعل بين الترمنال، الباش، والتطبيقات

هناك فرق بين الترمنال، الباش، والتطبيق المفتوح في الترمنال. الترمينا يُفتح منها الباش، ومن الباش تُفتح التطبيقات الأخرى. الترمينا وظيفتها عرض النص فقط. للتوضيح، إذا فتحت ترمينال، ثم نفذت الأمر `ls /dev`، ثم فتحت محرر نصوص مثل `vim`، ستلاحظ أنك تستطيع التمرير لأعلى ولأسفل في الترمنال، سواء كنت داخل محرر النصوص أم لا.

هذا يعني أنك، داخل محرر نصوص، لا زلت تملك وصولاً للترمينا، وتستطيع استخدام اختصارات لوحة المفاتيح الخاصة به. ولكنك لا تملك وصولاً مباشرًا للباش. لذلك، لا يوجد تعارض في استخدام Ctrl مع محرر النصوص.

اختيار الكي مودي فايرز للترمنال

لذلك، عند اختيار الكي مودي فايرز للتفاعل مع الترمنال، نختار:

- **Windows Key:** للتفاعل مع مدير النوافذ (DWM)
- **Ctrl:** للباش والمحطة (بشكل عام)
- **Alt:** هذا خيار ليس خاطئًا، ولكنه ليس مثاليًا بالنسبة لي. أجد صعوبة في الوصول إليه عندما أكون على الصف الرئيسي (الهوم رو). هذا أمر شخصي.
- **Ctrl + Shift:** هذا هو اختياري النهائي للكي بايندنز. أجد الوصول إليه أسهل.

الكيبورد ISO

هناك نقطة مهمة: أنا أكره لوحات المفاتيح ISO لأن موقع مفتاح Ctrl و Shift يكون قريبًا جدًا من بعضه، وهذا يسبب لي مشكلة. هذا تفضيل شخصي.

كي بايندنز إضافية

الكلام الذي ذكرته سابقًا هو عن الكي بايندنز العامة. أما الكي بايندنز الخاصة بالتطبيقات فسأتركها للفيديو القادم. في الغالب، أستخدم Windows Key معها أيضًا. أحيانًا أستخدم Ctrl مع بعض الأوامر، لكنني أتجنب استخدام الحروف في هذه الأوامر لتجنب التعارض.

خلاصة

أتمنى أن يكون هذا الشرح واضحًا. الهدف الأساسي هو فهم التفاعل بين النوافذ، الترمنال، والباش عند تصميم اختصارات لوحة المفاتيح. لا تستخدم نفس الاختصارات لكل شيء.

تهيئة LDWM و ST

السلام عليكم، في هذا الفيديو، سنقوم بإعداد LDWM و ST. الفيديو سيكون قصيراً جداً لأن كل التهيئة التي سنقوم بها ستكون داخل ملفات التهيئة. سنقوم فقط بتعديل القيم أو الخيارات التي نحتاجها. هذا طبعاً لأننا قمنا بالفعل بعمل كل التصحيحات اللازمة في فيديو سابق. سنستثني من هذا الفيديو الحديث عن الخطوط والألوان. موضوع الخطوط والألوان، وخاصة الألوان، يحتاج إلى شرح مفصل. لست متأكداً بعد ما إذا كنت سأتحدث عن الخطوط والألوان بتفصيل أم لا. إذا حدث ذلك، فسيكون ذلك غالباً في فيديو أو اثنين في نهاية السلسلة. يمكننا الرجوع وإضافة هذه التهيئة هنا إن شاء الله.

نستخدم نفس الطريقة التي بدأنا بها السلسلة: سنقوم بتسجيل الدخول إلى ستارت إكس. عندما كنا نضيف الحزم، كنا نعيد تشغيل DWM باستمرار لنرى تأثير كل حزمة. هذا طبعاً غير عملي، لأنه إذا كانت لدينا بعض التهيئات البسيطة، لكن عددها كبير، فمر العملي إعادة التشغيل في كل مرة. لذلك، سنقوم بعمل كل التهيئات دفعة واحدة، ثم نعيد تشغيل DWM، ثم نقوم بتسجيل الدخول مرة أخرى إلى ستارت إكس لنرى النتيجة. سنلقي نظرة على DWM بعد إجراء كل التهيئات. سنستخدم ربطات المفاتيح الافتراضية. لدينا أيضاً السوبر. هذا طبعاً جزء من الأشياء التي سنغيرها.

أول شيء سنغيره إن شاء الله في التهيئات هو: أعتقد أننا نحتاج إلى تحديث. يمكننا عمل تحديث سريعاً. حسناً، دعونا نفتح محطة أخرى ونبدأ عملنا. لدينا ملف `src` و `dwm`. لدينا أيضاً نسخة احتياطية `config.h.original` من الحزم التي قمنا بتثبيتها في الفيديو السابق. ملف `config.h` هو الملف الذي نحتاجه أثناء عملية الترجمة. هذا الملف يحتوي على ملفات الرأس التي يحتاجها الكود. يعتبر هذا الملف قالب يحتوي على تهيئاتنا. سنقوم بعمل تهيئاتنا هنا، ثم سيتم نسخ محتويات هذا الملف إلى `config.h` من خلال `make`, ثم يبدأ المُترجم العمل بناءً على هذا الملف. سنفتح الملف أولاً. لن أتطرق إلى كل التهيئات، سأقوم فقط بإلقاء نظرة سريعة عليها، وخاصة الأشياء التي سنغيرها. هناك العديد من الخيارات. مثلاً، سمك حدود النوافذ البسيطة، إظهار/إخفاء الشريط العلوي، وضبط مكانه (فوق أو تحت). الخيارات البسيطة.

هناك خياران آخران، هما ليسا موجودين بشكل افتراضي، فهما يأتيان مع مُنشئ الحالات. أحدهما يسمح بتفعيل/تعطيل مُنشئ الحالات. ميزة جيدة. النقطة الثانية هي تحديد الفاصل بين النص وما هو موجود على الحافة. يمكنك تحديد الحرف. هذا الحرف هو الذي تستخدمه عندما تريد عرض الحالات. لن نتطرق الآن إلى المواضيع الخاصة بالخطوط والألوان، كما ذكرت سابقاً. إذا احتجنا إلى قيم أخرى، فسنجدها على النظام. هذا ما يتعلق بالخطوط والألوان. سنضيف هذا إن شاء الله. ستلاحظ أنك لست بحاجة إلى معرفة لغات البرمجة. ظهر هذا في السطر التاسع بسبب هذه القاعدة تحديداً. لن نتحدث عن موضوع المتغيرات هنا. هذا هو التركيب النحوي الذي يقول أن القيمة ستذهب إلى السطر التاسع. القواعد هي الأشياء التي تحتوي على بعض المعلمات أو الخصائص الموجودة في النافذة، أو التي تُرسل إلى النافذة، ونحن نتعامل معها بقولنا أن DWM يتعامل مع هذه النافذة بطريقة معينة. لفهم هذا الموضوع، قام مطورو DWM بوضع كل ما تحتاج إلى معرفته بالفعل لمساعدة إدارة هذه القاعدة أو أي قاعدة أخرى تريدها. كل ما تحتاجه هو تشغيل الأمر `xprop`. هذا أحد الأوامر أو الأدوات الموجودة في Xorg ويتم استخدامه لعرض خصائص النافذة. سنقوم بسرعة بذكر هذا وسنوضح بعض الخصائص التي تم إنشاؤها في القواعد. هذه خاصة بفئات النافذة. الخاصية الأخيرة هي خاصة بالعنوان. إذا تمكنت من الحصول على هذه البيانات أو السمات للنافذة أثناء التشغيل، أو إذا كنت تعرفها، فيمكنك إنشاء قاعدة معينة. سأفتح محطة أخرى، دعونا نغير هذا. دعونا نُمسح الشاشة. هذه هي `xprop`. ستلاحظ أنه لا يوجد شيء في السطر. إذا نقرت على أي نافذة أخرى، فستحصل على خصائص هذه النافذة. ستلاحظ هنا وجود شيء يسمى فئة النافذة و اسم النافذة. هذه القيم هي التي يمكنك استخدامها لتحديد السلوك. يمكنك تحديد مكان العلامة، يمكنك وضعها في المركز، يمكنك وضعها في...

أول شيء سنقوم به هو إزالة قاعدة فايرفوكس. لست بحاجة إلى قاعدة جيمب أيضاً. سأقوم فقط بإزالة هذا ووضع `null` بدلاً منه. أعتقد أنه يجب أن يكون هناك على الأقل قاعدة واحدة موجودة هنا حتى يعمل DWM. سأضيف قاعدة `null` لا تحتوي على أي شيء. سأجعل الواحد هنا صفرًا، لأنه إذا جعلته 1، فستكون جميع النوافذ التي سأفتحها في DWM طافية، وأنا لا أريد ذلك. هذا يحدد عامل القياس. إذا فتحت نافذة على شاشة أكبر قليلاً من الشاشة. هذا هو عامل القياس، يمكنك تغييره من هنا.

يمكنك تحديد عدد الشاشات. بشكل افتراضي، هناك شاشة واحدة فقط. عندما تفتح نافذة ثانية، فإنها تظهر على نفس الشاشة. يمكنك تغيير عدد الشاشات أثناء عملك على inter. يمكنني هنا إضافة شاشتين، أو هنا يمكن تقليل الشاشة. هذا هو الرقم الافتراضي. أتركه 1 وأضيف أو أخصم الشاشات حسب الحاجة. نافذة واحدة تكفي. هذا يتعلق بتخطيط النوافذ. الأول هو دائماً الافتراضي. لا يوجد شيء هنا، هذا يعني أنه سيكون طافياً، والأخير هو الوضع اليدوي. أنا دائماً أستخدم هذا التخطيط. هذا ما يتعلق بالتخطيط. سنغير الجزء الذي نحتاج إلى تغييره. master هو الإعداد الافتراضي. أنا أغيّره إلى 4، وهو عدد النوافذ في التخطيط. هذا أحد التغييرات المهمة. يمكنك النظر إليه بحرية. الشيء الوحيد الذي يهمني هو match. يحدد العلامة للنافذة. إذا كانت هذه النافذة هي علامة معينة، سأجعلها. بعد الترجمة، سيكون هذا هو عدد النوافذ. لقد قمت بإنشاء match وهو رقم آخر، على سبيل المثال 4. هذه النافذة انتقلت إلى العلامة 4. تحدد العلامة التي ستنتقل إليها. يمكنك النظر إلى الأمثلة الأخرى هنا كما تريد. لقد قمت بعمل match وهو مثلاً match 3 أو match 2. هذا يقوم بعرض التخطيطات. كما ترى، هذا match أو match على سطر واحد. إذا تذكر، عندما تحدثت عن match في الفيديو السابق، قلت لك أن هذا يأتي مع command launcher. هذا ما يُظهر قائمة السياق. ثم هناك بعض الأوامر مثل close, move, resize. هذه أسماء تستخدم في الوسائط عند تشغيل هذا الأمر. هذه هي نفس الأشياء، أنا لا أستخدمها. لكنها موجودة. يمكنك توسيع هذا إذا أردت. أنا لا أعتقد أن هذا مفيد، فهناك أدوات أفضل من ذلك.

سأقوم بإلقاء نظرة على الملف لأتذكر تهيئاتي. أول شيء، هذا الافتراضي خاص بقائمة السياق. لسنا بحاجة إلى تثبيت قائمة السياق، حتى لو قمت بعمل شيء هنا، لن يحدث شيء لأن قائمة السياق غير موجودة. إنه يفعل شيئاً غير موجود، مهمة قائمة السياق غير موجودة، لذلك سأتركه كما هو. هذا يمنع حدوث تعارض مع ما سأستخدمه لاحقاً في Xbps. هذا الافتراضي، هذا من قائمة السياق، ستلاحظ أن كل التهيئة التي نقوم بها هي كالتالي: الأشياء التي أستخدمها كثيراً، أضعها بدون shift. الأشياء التي لا أستخدمها كثيراً أضعها مع shift. هذا هو النظام الذي ستلاحظه في كل التهيئة التي نقوم بها. لذلك سأزيل shift من هنا، لأنني أريد enter لفتح الترميز. أستخدمها كثيراً، لذلك لا أريد shift. هناك شيء آخر أريد القيام به بسرعة قبل المتابعة. سنزيل هذا من هنا عندما يأتينا مع هذا. هناك العلامة. ستلاحظ بسرعة، العلامة. هذا الأمر أو ربط المفاتيح الذي يجعل الشريط مرئيًا أو مخفيًا، فهو موجود افتراضيًا، لكن يمكنك إظهاره أو إخفاءه. أنا لا أستخدم الشريط كثيراً، لذلك سأضعه مع shift + mask. لأنني لا أملك شيئاً مع هذا الأمر بشكل عام، أحاول قدر الإمكان وضع الأشياء التي أستخدمها بشكل متكرر بدون shift. أنا لا أستخدم الشريط كثيراً. دعونا ننتقل إلى focus. focus الطبيعي هو عندما يكون لديك أكثر من نافذة مفتوحة، تريد الانتقال بينها. هذا ما ينقل التركيز بين النوافذ. يمكنك الانتقال بين العديد من النوافذ، لذلك سأترك هذا كما هو. دعونا ننتقل إلى arrange. لقد قمت بوضع هذه الخيارات عندما قمت بتثبيت patch، لذلك أضعها هنا تتناسب مع ملفّي. الفرق ليس كبيراً. لكنني أضعها هنا حتى يسهل التعامل مع ملفّي. إذا تذكر، هذا خاص بوضع النافذة أسفل أو أعلى. يمكنني تحريكها هنا. أستطيع تحريكها، يمكنني تحريك النافذة في الشاشة، بشكل افتراضي، يقوم بذلك هنا. أنا أستخدم control هنا، سأضع shift + mask. سأضع shift + mask هنا أيضاً. هذا يتعلق برفع النوافذ في الشاشة، زيادة عدد النوافذ الرئيسية. كما ذكرت من قبل، يمكني هنا زيادة عدد النوافذ الرئيسية. سأأخذ أول نافذة في الشاشة، هذا الرئيسي. سأضعها في الأعلى. لأجل وضعها كرئيسي، سأفعل هذا. إذا أردت تقليلها، سأفعل هذا. بشكل افتراضي، يجمع الوضع والترتيب. أنا لا أستخدم هذا كثيراً، لذلك سأجعله مع shift + mask. سنقوم بعمل ok. هذه هي نفس ربطات المفاتيح، أنا لا أستخدمها. أنا أستخدم عادةً نافذة واحدة أو اثنتين، نادراً ما أملك أكثر من ذلك. أنا أستخدم عادةً نافذة رئيسية واحدة. لست بحاجة إلى هذه الميزة كثيراً، وإذا احتجت إليها، سأجعلها مع shift. هذا عامل القياس الرئيسي. يمكنك تغيير هذه المعلمة، إذا تذكر، كنا في الأعلى. دعونا نعود إلى الأعلى بسرعة. عامل القياس الرئيسي. إذا تذكر، كان أكبر من 50%. لكن يمكنك تغييره. يمكنك تغييره عن طريق تغيير حجمه أفقياً. يمكنك عمل هذا أو هذا، لذلك يمكنني استخدامه أكثر من مرة. أنا أستخدم resize كثيراً، لذلك يمكنني تركه بدون shift. إذا فتحت نافذة أخرى، سأحتاج إلى تغيير الحجم حسب المهمة، لذلك سأتركه كما هو. هذا zoom. ستلاحظ هنا أنه قاتم. يفتح المحطة مع shift + return. أنا أغيره فقط. سأشرح ما يحدث هنا. إذا فتحت نافذة أخرى، وأردت جعلها رئيسية، بشكل افتراضي، يقول enter ليجعل النافذة الرئيسية. shift + enter يفتح المحطة. أنا أغير هذا. أتمنى أن يكون هذا واضحاً. سأفعل هذا. shift + mask. ستلاحظ أنني لا أقوم بالعديد من التغييرات. سأقوم بتصحيح هذا إن شاء الله. سأقوم بعمل هذا في الفيديو التالي إن شاء الله. سأقوم بعمل بعض التهيئات. ستلاحظ أنني أبطئ نفسي قليلاً. هذا موجود. هذا zoom. لا يوجد شيء آخر. tab view. هذا مع tab. هذا مع tab. إذا كنت تعمل في علامتين، تريد الانتقال بينهما. أنا أستخدم هذا كثيراً. دعونا ننتقل إلى هذا. الشيء الوحيد الذي أغيره هنا

هو q. أنا لا أريد q، أريد Q أكبر. سأغيره إلى q. أنا أغيره إلى q. أنا أستخدم shift. أنا أغير هذا لـ shift. أنا لا أستخدم هذا كثيراً. أنا أستخدم عادةً control + d لإغلاق النافذة، لكنني أريد وضعاً يُغلق النافذة ويُظهر الخطأ. سنتحدث عن هذه الثلاثة لاحقاً. أنا لا أفعل أي شيء معها. إذا تذكر، كان لدينا هنا Alt + T. انظر إلى الجزء العلوي. Alt + T أو mod + T يجعلك في وضع التجميع. M يجعل وضع Monocle. أنا لا أريد هذا. أنا لا أستخدم هذا تقريباً أبداً. سأضع هذا مع shift mask. سأجعل caps lock يعمل هنا. سنحرك هذا. هنا، أنا لا أغير الكثير، لذلك أضعها في shift mask. لا أريد هذا. حسناً. أحد الأشياء التي سأزيلها هو مساحة التبويب. هذه الأسطر سأزيلها. أنا لا أريد هذا. أنا لا أريد هذا، لذلك سأزيل هذين السطرين. إذا كنت تستخدم وضع الطافي، فيمكنك ترك هذا، أو إذا أردت الانتقال بين التخطيطات. أنا لا أحتاج إلى هذا. لذلك سأزيل هذين السطرين. هناك أيضاً هذين السطرين. سأريك السلوك. السلوك غبي نوعاً ما عندما أخطئ. إنه يقوم بعمل وضع صفر أو وضع shift صفر. إنه يقوم بعمل حركة غبية. إذا قمت بعمل وضع صفر، الذي هو حالياً Alt + 0. ستلاحظ أنه يُظهر جميع النوافذ التي فتحتها ويُظهرها في العرض. انه يحدد جميع العلامات، فهو يُظهر كل ما هو موجود في كل العلامات. أنا لا أستخدم هذا أبداً، وأنا لا أغير علامة النوافذ. بالنسبة لي، هذه الميزات ليست مفيدة، لذلك سأزيلها. لست أزيلها لأنني لا أريدها. أزيلها لأنني أحياناً، إذا أخطأت، فإنني أذهب إلى الصفر. أنا لا أريد أن يحدث هذا. لذلك سأزيلها لأنها تُغير العرض بطريقة لا أريدها. إذا ضغطت على الصفر عن طريق الخطأ، سيكون هذا موجوداً. سأترك هذه كما هي. هذا عندما يكون لديك أكثر من شاشة. إذا كان لديك أكثر من شاشة، فهذا الأمر وذاك الأمر يقومان بالتبديل بين الشاشات. إذا أردت تحريك النافذة المفتوحة من شاشة إلى أخرى، mod + shift هذا أو هذا، يُحدد نافذة أخرى من شاشة إلى أخرى. لن نتحدث عن هذا، لا يوجد أي تعديل آخر. التعديل الأخير الذي سنقوم به هو هذا. هذا ما يُغلق DWM. ستلاحظ أنني قمت بالفعل بعمل هذا، وهو mod + shift + q. ستلاحظ هنا mod + shift + q. mod + shift + q يُغلق النافذة. أريد هنا mod + shift + control + q ليُغلق control mask. M سأفعل هذا حالياً. لقد انتهيت. في الجزء السفلي، هناك شيء عن الماوس، لا نحتاج إليه. هذه هي كل التهيئة التي قمت بها في DWM. الشيء الوحيد الذي سأحتاج إلى الرجوع إليه هو الخطوط والألوان، لأنني تركتها هنا، وعندما نضع قاعدة للنافذة. بخلا هذه هي تهيئتي في DWM. لقد قمت بالفعل بحفظها. يمكنك إدارة نوافذك بشكل أفضل بهذه الطريقة. أنا فقط أعمل هنا. سنسخ التهيئة. إذا لم يكن لدينا أي مشكلة في التهيئة، فينبغي أن تكون التهيئة هنا. سننهي هذا، وسنخرج. Alt + Shift + Q يخرج من ستارت إكس. دعونا نجرب هذا. mode + enter يفتح نافذة. mode + enter مرة أخرى تفتح نافذة أخرى. لقد انتهى الأمر، لا شيء آخر. يفتح نافذة أخرى. نفتح نافذة أخرى. هنا، سنذهب إلى الأعلى. لقد قمت بتغيير هذا من mode + control إلى shift + i و j. لقد انتهى الأمر. كل تهيئاتنا تعمل بشكل صحيح.

دعونا ننتقل إلى ST. سنفتح نافذة أكبر قليلاً. دعونا نزال هذا. دعونا نزيل هذا. دعونا نترك هذا جانباً. أنا أملك caps في هذا الملف أيضاً. سنتحدث عن هذا لاحقاً إن شاء الله. دعونا نزال هذا. قبل أن ننسى. دعونا نفتح .Xresources. مخطط الألوان desert. دعونا نفتح ST. هذا الخط، سنغيره لاحقاً. الحجم. دعونا نقوم بعمل شيء صغير هنا. لأنني أقوم بالتكبير كل فـ دعونا نزيد هذا. دعونا نضيف 8. دعونا نجعل الخط 10. أنا لا أعرف كيف سيبدو هذا. لكن دعونا نجرب. أنا أملك خطاً آخر، لكن ليس بهذا الشكل. سأفعل هذا لاحقاً. الشيء الثاني، أرى هنا ST. أنا هنا أقوم بعمل تهيئة في ST. أنا قمت بعمل تبديل عشوائ للمسافة بين حدود المحطة والنص. أريد بعض المسافة. سأضيف هذا. دعونا نضيف 8. دعونا نضيف 10. أنا أملك 10 بالفعل في الخط. هذا تعليق هنا. أريد قراءته. لن أغير أي شيء هنا. يمكنك تغيير هذا كما تريد. شكل المؤشر، خط المؤشر. يمكنك تغيير هذا كما تريد. هذا الجزء الخاص بالألوان. لن أتحدث عن هذا الآن. لكن بشكل عام، مخطط ألوان المحطة مهم. ألوان المحطة أهم من ألوان DWM. ألوان المحطة هي في الأساس شاشة عرضك. عندما ترى مخطط الألوان الخاص بك على كل الشاشة. يمكنني قول هذا الآن. موضوع تصميم مخطط الألوان طويل. سأذهب إلى الأسفل قليلاً. أنا لا أتذكر تهيئتي. لقد قمت بتعديل شيء ما هنا. هناك المؤشر. شكل المؤشر. إذا أردت تغيير شكل المؤشر. الأعمدة والصفوف الافتراضية. هذا جديد. دعونا ننتقل إلى الجزء الخاص بالماوس. لا أحتاج إلى أي شيء هنا. دعونا ننتقل إلى هذا الجزء. هذا سيظهر هنا. لن أغير هذا الآن. هذا هو كل ما يتعلق بهذا الجزء. هذا الجزء هو الذي يغير الوضع الافتراضي، وهو alt. سأتركه alt لأنني لا أستخدمه على الإطلاق. الشيء الوحيد الذي أستخدمه هو الموضع الثاني أو المجموعة التي تسمى .terminal mode control + shift. لقد أتى مع st وسنتركه. إذا تذكر، كتبت أنني أستخدم دائماً control + shift مع أي شيء يتعلق بالمحطة حتى يكون قريباً من control. لا يوجد شيء أضفته، لا أذكر أنني قمت بتعديل أي شيء. ربما لا يوجد أي شيء. هذا الجزء يقوم بالتكبير والتصغير. الموضع هو control + shift. أنا لا أستخدم

priorities. أنا لا أغير هذا. هذا دالة التكبير. أنا لا أغير هذا. أنا أقوم بعمل هذا فقط لـ youtube، لأنني أعرف أن هذه الدالة تقوم بتصغير قليلاً في youtube. لأنهم يستخدمون شيء مختلف. هذا موجود بالفعل. لا أحتاج إلى إجراء أي تعديل. هذا نفس النظام، يعيد التكبير إلى الإعداد الافتراضي. دعونا ننتقل إلى هذا الجزء. هذا الجزء مهم. هذا ليس مهماً. دعونا ننتقل إلى ما يستخدم هذه الميزة. هذا copy و paste. هذا control + c و control + v. هذا واضح. control + c و control + v. هناك شيء مهم جداً يجب أن تخبره، الحروف الكبيرة مع مجموعة أوضاع المحطة يجب أن توضع هنا لأن وضع المحطة يأتي مع shift. يجب أن تقول أنه كبير. إذا جعلتها صغيرة، فلن تعمل بشكل صحيح. هذا الكابيتال هنا. دعونا ننتقل إلى make. هناك طريقان يمكن أن يفعلا نفس الشيء. يمكنك القيام بذلك هنا. لسنا بحاجة إلى شرح ذلك الآن. إذا كان لديك أكثر من لوحة اقتصاص، هناك لوحتا اقتصاص في لينكس. لسنا بحاجة إلى شرح ذلك الآن. هذان يقومان بنفس الوظيفة. عادةً في لينكس، هذا هو المعيار. shift + insert. سأترك shift + insert كما هو. هذا control + shift مع select، سأزيل هذا. ستلاحظ وجود شيء يسمى select و clipboard. هذا مكتوب هنا. هناك أكثر من لوحة اقتصاص. أنا لا أريد الدخول في التفاصيل الآن. لذلك سأزيل هذا لأنني قمت بالفعل بعمل لوحة الاقتصاص مع control + shift. shift + insert، كما قلنا، هو معيار في معظم أنظمة لينكس. لذلك سأتركه كما هو. لقد انتهينا. هناك شيء صغير آخر سنضيفه. شيء سنعدله وشيء سنضيفه. إذا تذكر، كان لدينا scroll up و scroll down. بشكل افتراضي، كان scroll up و scroll down مع shift + mask. أنا لا أفعل ذلك. أقوم بعمل هذا مع وضع المحطة. وضع المحطة. هذا control + shift. هذا مع page down. control + shift + u، scroll up مع d. أنا أستخدم control + u. أنا أستخدم control + d. نفس النظام في المحطة، لكن بدلاً من control، هو control + shift + d، control + shift + u. control + shift. هذه هي الفكرة. أنا أفضل استخدام وضع المحطة بدلاً من alt. هذا يجعل استخدامها أسهل، control + u. أنا أريد shift. نريد عمل شيء آخر هنا. هذا هو essential key. هذا هو k. هذا هو j. دعونا نقوم بعمل هذا. st. هذا k. هذا j. هذا st. لماذا st. cortex. عندما نقصد full page أو half page. أنا أفضل full page على half page. يمكننا رؤية ذلك لاحقاً، أو scroll up one line و scroll down one line. هذا كل شيء. هناك بعض التهيئات في الأسفل. هناك أشياء تتعلق بالماوس. أنا لا أستخدمها. يمكنك تغيير هذا كما تريد. دعونا نرى إذا كان التثبيت يعمل بشكل صحيح. يمكنني الخروج. control + shift. أنا أتوقع، أنا أتوقع... حسناً، أنا أتوقع. أنا في st لم أقوم بأي تعديل على DWM. المفترض الآن، إذا فتحت محطة جديدة مع mode + enter. الحجم سيكون أكبر افتراضياً. هذا لم يحدث لأننا غيرنا ملف التهيئة. إذا كنت أستخدم dev, control + shift + a, j, k, u. لدينا d وهكذا. هذه هي تهيئتنا. حتى نرى compile مرة أخرى. لقد قمت بعمل compile للشفرة. لقد قمت بعمل control + shift + y. هذا compile. لقد قمت بعمل control + shift + p +. هذا paste. هذا اختبار. shift + insert يعمل أيضاً. هذا تقريباً معيار. هذا معيار في لينكس. هذا اختبار. هذا كل ما أردت قوله. هذه هي كل التهيئة. فقط موضوع الخطوط لم يُذكر.

ثلاث إعدادات مهمة في نظام لينكس

{السلام عليكم}

في هذا الفيديو، سنتحدث إن شاء الله عن ثلاث إعدادات أو ثلاث خطوات مهمة جدًا بالنسبة لي، وخصوصًا أول اثنتين منها؛ لأنهما مرتبطتان باستخدام لوحة المفاتيح. من يتابعني منذ البداية، يعلم أنني كنت أواجه مشكلةً في إعدادات الكابسلوك (Caps Lock). سنُصلِح هذه المشكلة في هذا الفيديو. لنبدأ مباشرةً.

سنستخدم نفس البيئة التي نعمل بها عادةً، سنفتح Terminal ونستخدم الأمر `setxkbmap`. هذا الأمر ليس موجودًا افتراضيًا، لذا سنقوم بتثبيته. بالمناسبة، هذا الأمر جزء من `xorg`، وليس هو الأداة الوحيدة التي يمكن استخدامها، لكنه الأداة الأنسب والأسهل استخدامًا في هذه الحالة. لأنه جزء من `xorg`، فلا داعي لاستخدام أدوات تابعة لجهات خارجية.

يمكنك فتح صفحة `man` الخاصة بالأمر لمعرفة الخيارات والتفاصيل الكاملة، لكنني لن أفعل ذلك هنا. يمكنك مراجعة التفاصيل بنفسك. أهم ما يهمنا هو الملف الموجود في المسار `/usr/share/X11/xkb/rules/`. داخل هذا المجلد، يوجد ملف يسمى `base`. دعونا نتجاهل الأرقام التسلسلية ونغير لون الخلفية إلى Desert.

الأمر `setxkbmap` لا يقتصر على تغيير خريطة المفاتيح، مثل الكابسلوك، بل يقوم بمهام أخرى كثيرة، منها إضافة لغات. سأقوم بذلك الآن. كل قسم من الأقسام التالية هو عبارة عن خيار، ثم الوسائط التي يأخذها.

- **القسم الأول:** تحديد نموذج لوحة المفاتيح (Layout). أنا شخصيًا لست بحاجة لتغيير هذا، إلا في حالات خاصة، مثل استخدام آلة افتراضية أو لوحة مفاتيح مختلفة. لوحة المفاتيح التي أستخدمها قياسية، وأغلب عملي باللغة الإنجليزية، لذلك لست بحاجة لتغيير هذا القسم.

- **القسم الثاني:** تحديد اللغات (Layout Option). هذا القسم مهم لتحديد اللغات التي نريد استخدامها. سأستخدم هنا الأمر `setxkbmap` مع الخيار `layout`، ثم أسماء اللغات المطلوبة: `US,AR`. هذا يعني أن لوحة المفاتيح ستحتوي على لغتين: الإنجليزية (US) والعربية (AR). افتراضيًا، ستكون اللغة الإنجليزية هي اللغة النشطة. يمكنك إضافة لغات أخرى بفصلها بفاصلة.

- **القسم الثالث:** تحديد المتغيرات (Variants). هذا القسم يُحدد متغيرات كل لغة. للغة الإنجليزية (US)، توجد متغيرات مثل `dvorak, colemak` وغيرها. أنا أستخدم `qwerty`، لذلك لست بحاجة لتغيير هذا القسم. أما بالنسبة للعربية، يوجد متغير `digits`، والذي يُستخدم لعرض الأرقام بالشكل الشرقي. لإضافة هذا المتغير، سأضيفه بعد فاصلة: `,digits`.

لذا، سأستخدم الأمر التالي: `setxkbmap layout:us,ar variant:us,ar:digits`

هناك قسم آخر يتعلق بالكابسلوك. أنا شخصيًا لا أستخدم الكابسلوك كـ "كابسلوك"، بل أستخدمه كمفتاح آخر. لذلك سأستخدم `grp:ctrl_alt_shift_toggle`.

لذلك، سيكون الأمر الكامل كالآتي:

```
setxkbmap layout:us,ar variant:us,ar:digits
grp:ctrl_alt_shift_toggle
```

هذا الأمر سيغير إعدادات لوحة المفاتيح بشكل دائم. ولكن، هذا التغيير مؤقت، للحصول على تغييرات دائمة، يجب إضافة هذا الأمر إلى ملف `.xinitrc` أو ملفات بدء التشغيل الأخرى.

إعدادات بدء التشغيل (Startup)

لتشغيل هذه الإعدادات تلقائيًا عند تسجيل الدخول، يجب وضع الأمر `setxkbmap layout:us,ar variant:us,ar:digits grp:ctrl_alt_shift_toggle` في ملف بدء التشغيل المناسب. هذا الملف غالبًا ما يكون `.xprofile` في مجلد المستخدم الرئيسي.

إعدادات Vim

هناك بعض الإعدادات المهمة لـ `Vim` تُحسّن تجربة الكتابة، وخاصةً عند التعامل مع الأوامر الطويلة. أهمها استخدام أوامر التنقل (مثل `hjkl`) بشكل فعال، وإضافة بعض الاختصارات.

استنتاج

لقد قمنا بتغيير إعدادات لوحة المفاتيح، وعالجنا مشكلة الكابسلوك، وأضفنا بعض الإعدادات لـ `Vim`. آمل أن يكون هذا الفيديو واضحًا ومفيدًا. {سلام عليكم}

dmenu: شرح مفصل و أمثلة عملية

سلام عليكم، في هذا الفيديو، سنتحدث إن شاء الله عن dmenu. dmenu أداة مميزة في يونكس، تتمتع بخصوصية معينة؛ فهي تعتبر مُفسّر (interpreter) لواحد من أهم مفاهيم يونكس: الأنابيب (pipes) أو pipelines.

الفكرة ببساطة هي أنك تمتلك برنامجًا بسيطًا يؤدي وظيفة محددة، وينتج مُخرَجًا (output). هذا المُخرَج يُؤخذ كمدخل (input) لبرنامج آخر بسيط، وهكذا. يعمل كل برنامج على جزء من المهمة، وتتضافر هذه البرامج مع بعضها لتحقيق مهمة أكبر.
dmenu يُسهّل هذا الأمر بشكل كبير.

وظيفة dmenu:

dmenu يأخذ قائمة من السطور (lines) كمدخل، ويعرضها في قائمة (menu). ثم تختار من هذه القائمة، ويظهر اختيارك كمخرج. الفكرة بسيطة جدًا ووظيفتها بسيطة، لكنها قوية جدًا في نفس الوقت.

لنرى كيف يعمل هذا الأمر عمليًا: سنستخدم نفس البيئة التي بدأنا بها. سنفتح متصفح فايرفوكس، ثم نذهب إلى موقع ساكلس (suckless.org). يمكنك تنزيل dmenu من مدير الحزم (package manager)، ولكن ساكلس يعتبره أداة مستقلة، لذا سنقوم بتنزيله وتثبيته من ساكلس. كود المصدر متوفر في قسم "tools"، على عكس dwm و st اللذين يوجدون في قسم "home". dmenu موجود في قسم "tools". انتقل إلى صفحة dmenu واكتب الأمر git clone كما فعلنا سابقًا.

سنقوم بتنزيل dwm ،dmenu، و st. أنا لا أقوم ببناء حزم (packages) مطلقًا لـdmenu. أحتاج فقط إلى إدخال قائمة خيارات، ويظهر لي اختيار واحد فقط كمخرج. أريد أن أبحث داخل الكود.

يقول البعض أن dmenu هو launcher فقط. هذا غير صحيح. launcher هو مجرد تطبيق مبني على dmenu. لكن dmenu نفسه هو الذي يقوم بالوظيفة الرئيسية: يأخذ إدخالًا، ويعرضه في قائمة، وتختار خيارًا واحدًا منه كمخرج.

فكرة launcher هي تجميع لعدة سكربتات مع dmenu. هذه السكربتات هي التي تحتوي على وظائف launcher. لتوضيح الأمر مجددًا: dmenu هو dmenu. هناك برنامج آخر اسمه dmenu_run الذي يطبع الخيارات الموجودة في مسار معين. هناك أيضًا سكربتات أخرى موجودة في مسار محدد يستخدمها dmenu_run ليعطيك الخيارات المرغوبة.

مثال عملي:

لنقل أن لدينا قائمة: أحمد، محمد، علي. هذه قائمة بثلاثة عناصر، يمكننا عرضها باستخدام dmenu. ستظهر القائمة، ويمكنك التنقل فيها باستخدام مفاتيح التحكم (Control + n, Control + p) أو أسهم لوحة
المفاتيح.

الاستخدام الأساسي هو الكتابة، فيقوم البرنامج بفلترة القائمة بناءً على ما كتبته، ويظهر لك الخيارات التي تتضمن ما كتبته. إذا لم أرغب في خيار به حرف "م" مثلاً، فسيظهر لي فقط "أحمد". لاحظ أنه سيظهر "أحمد" مع "محمد" لأن كلاهما يحتوي على حرف "م". إذن، المدخل هو القائمة، وdmenu يعرضها كقائمة، ثم تختار خيارًا واحدًا.

ملاحظة مهمة:

dmenu لا يحتوي على خاصية البحث الضبابي (fuzzy finder) افتراضيًا. إذا كنت ترغب في استخدام خاصية البحث الضبابي، فيمكنك تنزيل الباتش المناسبة. أنا شخصيًا لا أستخدم dmenu كثيرًا، سأتحدث عن هذا لاحقًا.

أمثلة متقدمة:

مثلاً، إذا كتبت "أ" ثم "م"، فسيظهر لك "أحمد" و "محمد". إذا كتبت "م" فقط، فسيكون لديك "محمد" و "أحمد". تختار من الخيارات المعروضة. يمكنك أيضًا إضافة سطر جديد في النهاية، مثلاً "الخروج" مع قائمة الخيارات.

أهمية هذا الكلام هي إمكانية أخذ المخرج الذي اخترته، وبدلاً من طباعته، يمكنك تنفيذ إجراء بناءً على اختيارك. مثلاً، يمكنك إنشاء متغير اسمه `choice` ويأخذ قيمة المخرج، ثم بناءً على قيمة هذا المتغير، يتم تنفيذ سكريبت معين. مثلاً، إذا اخترت "محمد"، فيتم تنفيذ سكريبت معين.

يمكنك إنشاء برنامج صغير للتعامل مع هذا المخرج. مثلاً، إذا اخترت "محمد"، يتم طباعة رسالة معينة.

مثال مع `sxhkd`:

`dmenu` يصبح أكثر قوة مع برامج ربط مفاتيح (keybindings) مثل `sxhkd`. في هذه الحالة، لن تحتاج إلى فتح `dmenu` يدويًا، بل يمكنك ربط اختصار لوحة مفاتيح معين بتشغيل `dmenu`. مثلاً، يمكنك ربط `Super + Shift + Backspace` بتشغيل `dmenu`.

الخلاصة:

`dmenu` أداة قوية ومرنة. يمكنك استخدامها لإنشاء قوائم تفاعلية، وإجراء عمليات بناءً على خيارات المستخدم. أنا شخصيًا لا أستخدم `dmenu` كثيرًا، وأفضل استخدام `fzy` لأغراض البحث، ولكن `dmenu` يبقى برنامجًا رائعًا وله استخدامات متعددة.