Modernizing Accessibility for Desktop Linux

Matt Campbell

GNOME Foundation

Accessibility overview

- Assistive technologies (AT): screen reader, alternative input, etc.
- IPC between AT and applications
- Accessibility tree
- Events: focus change, caret movement, etc.

Free desktop accessibility history

- Assistive Technology Service Provider Interface (AT-SPI)
- First implemented by Sun in 2001 atop CORBA
- Ported to D-Bus starting in 2008
- Under-resourced for 10+ years

Problems with AT-SPI

- Disconnected from windowing system
- Assumes trusted apps, e.g. in focus event handling
- No support for sandboxing, e.g. Flatpak

A deeper problem: chatty IPC

- AT-SPI requires its own D-Bus message bus
- Other platforms have this problem to varying degrees
- Latency of round trips adds up
- Partially mitigated by caching
- Often bottlenecked on the app's main thread
- One cause of unequal access for blind users

Introducing the Newton project

- Wayland-native accessibility
- Named after home of Carroll Center for the Blind
- Designed for untrusted apps
- Apps push serialized accessibility trees and updates

Why push?

- Minimizes latency
- More resilient to app hangs and busy main thread
- Proven in Chromium and Firefox internal IPC
- Some drawbacks, e.g. performance in large documents
- Enables exciting possibilities (more later)

High-level design goals

- No impact when ATs aren't active
- Compositor is the final source of truth
- Shift as much complexity as possible to ATs

Building on AccessKit

- Cross-platform accessibility infrastructure in Rust
- Already implemented for Windows, macOS, and AT-SPI
- Internally uses push approach; serializable tree updates

Current status

- Protocols not finalized yet
- Prototype implementations in AccessKit, Mutter, and Orca
- Can demo end-to-end with some Rust GUI apps
- Integrating AccessKit into GTK

What's next

- Finish GTK integration
- Measure performance; optimize where needed
- Implement for GNOME Shell UI
- Finalize protocols
- Merge into upstream projects (Mutter, GTK, Orca)
- Documentation

"Show me the code"

- Orca prototype branch: https://gitlab.gnome.org/mwcampbell/orca/tree/newton
- GTK AccessKit integration: https://gitlab.gnome.org/mwcampbell/gtk/tree/accesskit

Looking ahead: new possibilities

- Accessible remote desktop with no AT on the remote machine
- One-to-many use cases: accessible screenshots and screencasts

Conclusion

- The overhaul that free desktop accessibility needs
- We can advance the state of the art in accessibility

Special thanks

- Sovereign Tech Fund
- GNOME Foundation

Email: mattcampbell@pobox.com
Mastodon: @matt@toot.cafe