The text-based environment for blind persons: conception and operating system design

Michael Pozhidaev
michael.pozhidaev@gmail.com

International research journal, 2013, issue 2, pp. 63-66
(ISSN 2227–6017)

The paper covers conclusions of the attempt to create a prototype of text-based user environment without graphical objects and operating system based on it. The acquired results prove the conception is suitable for handy device for wide range of blind users. The technical details for further research are suggested.

Keywords: accessibility, blind persons, GNU/Linux, Java, operating systems.

Introduction

Various information technologies became an essential part of social and professional life, but level of their accessibility for blind and visual impaired persons remains still insufficient. Although some results and experience in this area are acquired so far, blind and visually impaired users are not able to feel themselves completely free using PCs and mobile devices. That fact can be treated as one of the reasons tackling high integration of disabled people into social life.

Currently the popular way is to install some screen reading software, such as Jaws for Windows by Freedom Scientific [11], VoiceOver by Apple Inc. [2], as well as Orca [15] — the favourite solution of GNU/Linux users. This software takes the information on screen and transforms it into speech form, describing any action user does. Today there is a ubiquitous Graphic User Interface (GUI) almost on every computer and it is the most important problem we should take into account here. This type of user interface (UI) was designed to be controlled mostly by mouse, as well as other pointing devices. This input method is rather convenient for sighted users, but for blind persons it is not the case. They are enforced to navigate over graphical objects on a screen, such as window widgets, menus, dialogs by keyboard only and this way takes a lot of extra time to do. The mentioned way can be considered as appropriate for work at home or at office, but in crowded and noisy environment, such as at international airport, show or conference, it is turned out completely unacceptable. In addition, running OS with full GUI takes more hardware resources than it is needed for accessible solution itself. This paper offers an accessible OS of a new design based on GNU/Linux and covers the experience already got by creating preliminary prototypes.

1 First prototypes

The work on the initial prototype was launched in 2008 as an attempt to create light accessible distribution for blind users [21, 22, 23, 24]. It uses GNU Emacs [8] as a main user environment with added special extension by T. V. Raman from Google Inc called Emacspeak [6]. The system is based on the Linux kernel and is applicable for installation without any external sighted help, except of boot device selecting in BIOS setup utility, which does not use any features from external software and evidently is not compatible with any speech-based accessibility. The installation process is performed by copying live-CD environment to a hard drive as it is with several minor fixes. We will not describe this technique in details here, since it is widely popular and well-known approach. The only important thing is that cloning process doesn't require any UI and can be done by single command launch with provided set of necessary parameters.

The environment was enhanced with some additional software developed by the authors as part of work on the system being described. This software includes tools for text books listening with text-to-speech (TTS) engines, main menu plug-in for quick applications launch, media player control service (will be described later), the set of auxiliary scripts and so called speech server. The speech server is needed to manage stream of speech commands in real-time mode and to prevent simultaneous speech signals overlapping. It doesn't synthesize speech by itself and uses external TTS engines, in particular, RHVOice [16].

Although this distribution failed as instrument for wide range of users, it brings important experience, proving the general conception is right. As known, GNU Emacs aims to be a flexible text editor but during developing process list of its features went out of editor purposes. Now it contains file manager, mail and news reader, calendar, FTP client and very restricted web browser. The crucial advantage allowing consider it as an accessible environment is an ability to bring to user every working object in text form. For example, mail message can be easily constructed as text file with recipient address on the first line, the subject on the second and with message content on all others. Nearly all needed operations could be reorganized this way if they don't imply graphic materials. GNU Emacs is also capable with showing web-pages in text form, providing content as one text document and excluding pictures. While user navigates over prepared text object, the speech extension listens all cursor movements and forwards stream of commands to speech server for further translation into audio form. If user moves from one text line to another he hears line text under new position and hears new character in case of moving left-right inside of one line.

This system became a working platform for its developers for several years and it makes possible to get some conclusions about its advantages and disadvantages. The main advantage is a a very high speed of working process. With some user training, speed approaches to speed of usual sighted work. It can be explained by user interface simplicity, since it doesn't have any GUI elements. A user doesn't take care

about what objects he has on a screen and how they are organized. Even more, any association between objects and their graphic representation is not required at all. In addition, light environment yields low system resources consumption and it makes possible using of cheap mobile devices. One of the installations was performed on ASUS EEE PC 1025C netbook ($330 in Amazon shop) and it was successfully used in travel and during conferences participating. That experience implies tasks successfully done during flights and work inside of crowded and noisy airports, including three the busiest airports in Europe (London Heathrow, Paris Charles de Gaulle and Frankfurt-am-Main) and three international Moscow airports. The set of performed tasks covers maintaining wifi-connection for mail and news reading, handy notes, mp3 and text books reading and rarely voice recording.

Actually, a wide range of accessible tasks can be easily explained by large variety of software available among GNU/Linux distributions, but some additional possibilities should be mentioned especially detailly. There are two graphic software packages: Latex [12] and Lilypond [13], capable with taking input material in form of text files. Latex is a publishing system for physics and mathematics books or papers and Lilypond is a music score typesetter. Both of them gives a very high quality of an output and remains accessible for blind users. Known experience proves they are a real way to create materials even at scale of thesis. Final sighted checking of an issue is still required but work is done mostly by blind persons on their own. That software is a very special case due to high level of required user skills but anyway we would like to mention it. Latex can also produce presentations in pdf-format (proved by known experience of blind users), can include graphic figures constructed by text form commands and embed music objects from Lilypond.

We took a look what this system offers, but more important is the set of discovered disadvantages, making prepared system inappropriate to be used widely. The developers team is about to give up any active work on GNU Emacs approach due to following reasons:

1. Weak protection against improper user actions. Even a very easy inaccurate step breaks proper system behaviour. GNU Emacs completely relies on user awareness what he does.

2. The platform is not suitable for developing complex user applications. The GNU Emacs is a Lisp run-time environment purposed for managing a set of text areas. All of such areas called "buffers" are linked weakly with each other and any attempt to create an application with more than one buffer gives generally unstable product with a lot of glitches. The system must be capable of quick extensions creating because user needs various add-ons for social networks, blogs reading and access to other web-services.

3. Web-browser without Java Script support. Web-browsers became something grater than applications for HTML-pages viewing and may be described in turms of a platforms for launching

web-applications. GNU Emacs has a text-mode browser that can be used for HTML-pages parsing but nothing else.

4. User unfriendly interface. The environment of GNU Emacs can be considered suitable only for professional users with high level of experience in GNU/Linux. No national language support is available.

5. No support for some closed but freely available application such as Skype and maybe some others. Skype has an accessibility support but it cannot be covered by any GNU Emacs utilities.

6. No support for popular office documents formats.

Since a lot of free and open source software is available publicly, it should be involved in new accessible environment creation. In two sections below we will describe new UI approach and general design of accessible OS free from disadvantages listed above as far as it possible.

2 New user environment

New user environment should consist of following items on a screen:

1. A set of tiled working areas

2. A easy access command line for a quick operations launch

3. A popups support for dialogs and interactions

4. Main and context menus

Nearly all of these things came from GNU Emacs as they are, but they should be managed by general application mechanism, which GNU Emacs doesn't have. Every text area should be strongly associated with some application and must be closed on application shutdown. All text areas are displayed in tiled mode on the screen and currently active application should decide which of them and how must be shown. User should be allowed to switch from one application to another quickly (for example, by Alt+Tab key combination) with switching of all corresponding visible areas on a screen. One more important feature is a quick text search through text in any visible area without application support.

The term "text area" we used here implies rectangular area on the screen filled with some text shown by monospaced font. Text color and size should be available for changing easily by system-wide commands without any application support. On screen content must be capable of using by users with low vision (not totally blind). The low vision users can request a feature of highlighting with the easy grey background row and column, where cursor is placed.

A command line support plays highly important role in suggested approach. In contrast to widely popular GNU/Linux shell expressions, a command line, we are describing, should accept short words suitable for quick typing. Inside of noisy room it is easier to press Alt+x on keyboard to invoke command line, type "news", "mail", "message" or whatever else user wants to launch, than to look for required item in menus listening speech output. The keyboard com-

bination Alt+x is taken from GNU Emacs and, surely, it is really convenient way. We consider a command line here as particular case of "popup" area. Popup areas are text areas shown on screen at bottom for a short time and purposed for additional user interactions or conversations. Popup areas can be also applicable for various types of menus.

Java SE [10] is chosen as the main programming language for developing accessible environment itself, as well as for all user applications. There are a lot of well-known Java libraries available publicly, providing various functions needed for user applications. We mean here JavaMail [9] for mail reading, Rome [18] for RSS parsing, Apache POI [1] for office documents filters etc. A user interface layout for each particular application is not described here, since in most cases it is quite obvious. For example, mail reader should consist of three areas: mail groups, the list of mail messages in a group and selected message text. The file manager looks like usual twin-panel manager with additional area at a bottom of screen for list of active tasks (files copying, moving etc). General user navigation over text objects and whole system can be taken from GNU Emacs and left almost without any changes. It is a very good part of GNU Emacs legacy, except of one thing: spoken text must be constructed by application and it may be different than text on screen. In GNU Emacs user hears always a text on screen, gathered by emacspeak.

3 System core

We should take a close look at system core behind accessible UI. It consists of several components listed below:

1. An event queue launched inside of Java virtual machine with separate thread
2. An LDAP interface for flexible user data storing, like address book, calendar etc
3. D-Bus [7] for access to a number of system services
4. An auxiliary service for media player control
5. A speech server
6. A speech-enabled window manager for X.org
7. A small screen reader based on AT-SPI [3] services.

The event queue stands here for central dispatcher of events and the most important of them, of course, are user input commands. That queue should be also accessible for an events exchanging between several applications, being running in the environment. It is very important thing because this feature is the most proper way for multithreaded applications synchronization.

Now there is a general moving in GNU/Linux world toward D-Bus system services and that fact makes a lot of things easier for system developing. Using D-Bus, which has good support in Java, we can get a easy control over key system services, like Udisks [19] for removable drives, Network Manager [14] for network connections and

others. Systemd service [17] should be mentioned here as an especially hopeful suggestion, but needs more time for clarification whether Systemd is really able to become a reliable low-level component or not.

Media player service must take care of any user commands to listen music file or "speaking book". It should be controlled also through D-Bus and invoke real media player (very likely VLC [20]). Media player can not be launched directly, since blind user needs some features for "speaking books" (literature recorded by narrator as audio files) including bookmarks managing etc. In addition there is a special format of books for blind persons called Daisy [5] and it also should be supported at the level of media player control service.

4 A wWeb-browser and the AT-SPI applications

There is a couple of special cases when user has to deal with tradition GUI anyway :

1. A web-browser
2. Closed popular applications (e.g., Skype)

Without these applications support our system can not be considered full. In case of web-browser its nature doesn't allow express its functionality in any text only form. Skype, due to its license restrictions, should be taken as a whole binary application. Hence, we have to implement some things, making these application available for blind users, although both of them evidently are exceptions in proposed conception of text only interface. Fortunately, GNU/Linux GUI has additional service called AT-SPI, helping disabled persons operate with GUI applications and it remains still available for us. If we choose Firefox as suitable web-browser we can use it and Skype through AT-SPI functions. Regarding web-browser we should notice, to be fair, there is one more solution — ChromeVox add-on [4] for Chrome/Chromium browser, but its less comfortable because it generates speech by itself and doesn't expose any data outside of browser.

That means there is no way to pick up browser data from external tools, for example, to be copied into custom buffers, like clipboard. Since we rejected any existing GUI we can not use prepared screen readers used for transforming AT-SPI data into speech and have to implement its replacement.

The last thing that we need to get our design completed, is a tiny window manager. Main Java virtual machine we have described in previous sections and additional applications like Firefox or Skype are independent processes with their own windows on a screen controlled by X.org server. A special speech-enabled window manager should be created to link all of them into one whole suite. Java environment could be considered as main application playing predominant role. Other applications became available easily with simple way to switch between them with informative speech notifications. There is no need to develop such window manager from scratch since there are a lot of simple suitable examples.

Conclusions

The design described here is currently in active development and authors intent to create a prototype suited for illustration what such system could be used for (the project is designated as "LUWRAIN"). Meantime a lot of required components are already created since almost all of the tasks are the widely known problems solved by large variety of FOSS projects. The only thing should be written completely from scratch is the Java user environment and its conception is the main research goal.

A couple of related questions stays undiscussed with these paper. First of them is the braille display support. These theme doesn't need any research since Os just has to support corresponding devices. The second is a using of tablet computers. A tablets are a big problem, because this sort of devices can be used only through tracking graphical objects position on a screen. Anyway, this area needs a lot of deep research, respecting experience of the projects providing accessibility for multi-touch devices. It is especially important due to threat for low-cost laptops by tablets in current market situation. As a matter of fact, some companies are about to prefer tablets production rather low-cost laptops.

However, a lot of cheap embedded computers based on ARM architecture are coming and they should be considered as real companions for wide range of disabled persons. GUI doesn't play valuable role in their using and we may expect demand of any text-based solutions.

References

[1] Apache POI — the Java API for Microsoft Documents // http://poi.apache.org/

[2] Apple — Accessibility — VoiceOver — In Depth // http://www.apple.com/accessibility/ /voiceover/

[3] Assistive Technology Service Provider Interface // http://www.linuxfoundation.org/ /collaborate/workgroups/accessibility/ /atk/at-spi

[4] ChromeVox // http://www.chromevox.com/

[5] DAISY Consortium // http://www.daisy.org/

[6] Emacspeak — The Complete Audio Desktop // http://emacspeak.sourceforge.net/

[7] freedesktop.org — Software/dbus // http://www.freedesktop.org/wiki/ /Software/dbus

[8] GNU Emacs // http://www.gnu.org/software/emacs/

[9] JavaMail API // http://www.oracle.com/technetwork/ /java/javamail/index.html

[10] Java Standard Edition // http://www.oracle.com/technetwork/ /java/javase/index.html

[11] JAWS Screen Reading Software by Freedom Scientific //

http://www.freedomscientific.com/ /products/fs/jaws-product-page.asp

[12] LaTeX — A document preparation system // http://www.latex-project.org/

[13] Lilypond // http://www.lilypond.org

[14] Network Manager // http://projects.gnome.org //Network-Manager/

[15] Orca // https://live.gnome.org/Orca

[16] RHVoice // https://github.com/Olga-Yakovleva/RHVoice

[17] Systemd // http://freedesktop.org/wiki/Software/ /systemd/

[18] The Rome projects // http://rometools.org/

[19] Udisks // http://www.freedesktop.org/wiki/ /Software/udisks

[20] VideoLAN — Official page for VLC media player // http://www.videolan.org/vlc/

[21] Pozhidaev M. S. The description of ALT Linux based solution for the blind // Information technologies for the blind in the modern world. Problems and prospects: Proceedings of the 7th conference (December 4, 2008). — Moscow.: 2008. — pp. 82–86

[22] Pozhidaev M. S., Kamynin ‘A. N. Evolution of the environment for the blind ALT Linux Homeros // 7th conference of FOSS developers “On Trubezhe”. Pereslavl, June 26–27, 2010 Proceedings of the conference. — Moscow. : Institute of logic, 2010. — pp. 78–80

[23] Pozhidaev M. S. The distribution with accessibility technologies ALT Linux Homeros: the first experience // The 8th conference of FOSS developers: proceedings / Obninsk, July 25–26, 2011 Moscow.: ALT Linux, 2011. pp. 9–11

[24] Pozhidaev M. S. Developing and prospects of the distribution with accessibility technologies ALT Linux Homeros // Open technologies: proceedings of the 8th international conference “Linux Vacation / Eastern Europe” 2012, Grodno, June 07–10, 2012 / Ed. Kostuck D. A. — Brest: Alternative. — pp. 23–25