Improving Usability of the Linux Kernel Configuration Tools

Kacper Bak
Generative Software Development Lab
University of Waterloo, Canada
kbak@gsd.uwaterloo.ca

Karim Ali
PLG Group
University of Waterloo, Canada
karim@uwaterloo.ca

ABSTRACT

Tailoring a Linux kernel to one's needs has been one of the most cumbersome tasks a GNU/Linux user can do. There have been many attempts to overcome this problem by introducing smarter configuration tools. Those tools, however, still lack some important features, which discourages users from using them. In this project, we plan to address the problem of usability of the Linux kernel configuration tools. Our aim is to identify the major usability issues with current tools, propose a better user interface and evaluate it on a group of Linux enthusiasts.

Author Keywords
usability, linux, kernel, configuration

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Miscellaneous

INTRODUCTION

GNU/Linux is a free operating system with Linux as its kernel. The Linux kernel[1] is a mature and very complex piece of software. It has been developed by thousands of programmers led by Linus Torvalds since 1991. The kernel supports many computer architectures, network protocols, thousands of drivers, and many debugging options.

Structure of the kernel is very modular, so that each user can tailor it to her particular needs and specific hardware. Users upgrade their kernels mostly to improve stability, apply new bug-fixes, add functionality and drivers. Sometimes it is necessary to upgrade kernel in an embedded system, such as network router. Recent research [5] showed that the whole kernel is composed of almost 5500 features, of which 89% are user-selectable. Thus, the variation space is huge and the configuration process requires a broad computer knowledge. Users can configure kernel either by manually editing the .config file or by using some configuration tool (i.e. menuconfig, xconfig, gconfig). The first option is discouraged, because it is very error-prone due to lack of automated constraint validation/propagation. Using a configurator is a better solution, but it is still a laborious process.

[1]Available at: `http://www.kernel.org/`

PROBLEM

All the configuration tools are front-ends built on top of a single engine called Linux Kernel Configurator (LKC). LKC analyzes kernel variability and supports users during the configuration stage. LKC uses internally the KConfig language to represent variability and dependencies among features. As several researchers [6, 5] showed, there is a direct correspondence between well-understood feature models and practically crafted variability model used by the Linux kernel. In contrast with formal feature models, LKC is not supported by any formal reasoning engine (e.g. SAT-solver). This is a serious problem, because users can unconsciously create wrong configurations while having no clue why a particular configuration does not work.

Many users complain about the lack of kernel autoconfiguration [1, 7], where LKC could detect current hardware and automatically choose relevant modules. The situation is slowly improving as developers added the localmodconfig [2] target to Linux-2.6.32. The command detects current kernel configuration and applies the same options to the new kernel. It is a step ahead, but the tool is very simplistic and assumes that all the required modules are already loaded. For example, if a computer has built-in Bluetooth, but the module is not loaded, the new kernel will not support the Bluetooth device. Furthermore, the autoconfiguration script offers a very coarse level of options granularity, e.g. if a computer has one sound card, such as Intel HD Audio, the script will select all available sound card drivers and all Intel HD Audio modules. The autoconfiguration tool reads configuration of the running kernel instead of detecting the actual hardware, e.g. for a laptop with the Core Duo 2 processor, it selected 686 processor in the new kernel, because that processor is selected in the running kernel.

Linux kernel configurators are targeted to advanced users. Some kernel developers expressed opinion [3] that kernel should be configured only by experienced users. While this statement might be true, there are still lots of Linux novice users who want to learn how to configure kernel or need to compile a particular driver. As of now, there are millions

[2]More info at: `http://bit.ly/cPgq8R`

Submitted for CS 889 - Open Source Usability.

of web pages describing how to configure and compile the Linux kernel. Novice users are often overwhelmed by the number of available options and required knowledge. LKC offers no progressive-disclosure for them.

PAST ATTEMPTS

Some of the above issues were already solved in 2002 when Eric S. Raymond presented the CML2 configuration system [4]. It allowed for effective reasoning on kernel feature model and also provided progressive-disclosure. There was a long debate and flame war about using that system [2]. Finally, it was rejected for various (sometimes silly) reasons, such as Eric S. Raymond's attitude, Python dependency, complexity, radically new design. Many developers preferred to introduce gradual updates instead of applying one big patch.

SOLUTION

We would like to improve the usability of the Linux kernel configuration tools either by fixing them or by adding a more intuitive front-end. Many of the LKC shortcomings can be fixed incrementally, without throwing away the whole infrastructure. In our solution we would eliminate the problem of reasoning by limiting user-selectable set of features. A user should only see the most relevant and non-conflicting configuration options. Furthermore, the software should be aware of autoconfiguration limitations. Thus, the tool shall ask the user about non-discoverable devices. The final solution will depend on two initial studies on users.

In the first study people will be asked to use current tools (i.e. xconfig) to configure and compile the Linux kernel. We would like to observe how participants interact with these tools and what problems they have. This will create a baseline for comparing our new interface against. After that, we will propose several paper prototypes of user interfaces and apply the Wizard of Oz method to quickly evaluate and refine them. The method should capture users' expectations and reveal design faults in the proposed interfaces. Afterwards, we shall be ready to construct and implement the most promising solution.

We believe that the proposed improvements will make the kernel configuration infrastructure more intuitive and easier to use. Many of the techniques are not tied to this specific project. The problem of effective product configuration is very common, but rarely solved, it is enough to mention package managers, feature model configurators or even tailoring medical messages to specific patients. The FOSS community could use the ideas in any kind of software that has variability in it.

SCOPE

Initial User Studies

We are going to conduct the two initial studies mostly on computer enthusiasts who have some experience with GNU/Linux and are interested in kernel configuration. The studies will also include less experienced users, so that we can observe the most fundamental problems that people have with configuration. In both studies the number of participants will be around 10.

In the first study participants will be asked to configure the Linux kernel for a popular laptop for typical home/office use. A person succeeds if the new kernel is ready to play music and movies, connect to Ethernet, use WiFi, memory cards, USB devices, Bluetooth, CD/DVD.

In the second study we will propose several UI paper prototypes that address the problems found during the first study. We will present the prototypes to the participants and ask them how they would configure the kernel again using these prototypes. Observations from the second study will tell us which prototype is the most intuitive and what users expect from such a configurator.

Implementation

We will implement a graphical user interface in some popular language. The interface shall match expectations of our target user group. However, it will not try to solve problems of novice or advanced users. The interface will most likely have some connection to the backend, or will be able to update the .config file with user-selected features. When it comes to hardware, the interface will support the most common non-discoverable devices, such as wireless network devices, external devices, and sound cards.

EVALUATION PLAN

In order to evaluate the usability of our tool, we will conduct a user study on two groups of participants (group A and group B). Both groups will be presented with the same set of configuration parameters that need to be set for a Linux kernel. However, **Group A** will use the currently available configuration tools (e.g xconfig), while **Group B** will use our tool. At the end of the study, we will measure how close each group is to the sample kernel configuration that we presented to them. Consequently, we will be able to assess how usable our tool is compared to the current ones.

OUTLINE

Table 1 shows proposed outline.

Milestone	Start	End
Initial Proposal	May 27	May 31
Final proposal	May 31	June 14
First User Study	June 14	June 16
Second User Study	June 16	June 22
Implementation	June 23	July 9
Final User Study	July 12	July 14
Report and Blog Entry	July 15	July 28

Table 1. Project Milestones

DIVISION OF LABOR

Dividing the work in our project might not be as beneficial as working together especially for the user studies that we plan to do. For the studies, working together will allow us to observe more facts about the behavior of our participants and their reposonse to our questions. Dividing those studies between us will lead to either observing less facts if each one of us interviewd a group of users alone, or gathering incomplete facts if each one of us did part of the interview for all our participants. The only part of our project where dividing the labor makes sense is the implementation. However, we will not know how to implement our tool until we do our initial studies. Table 2 shows proposed division of labor so far.

Task	**Karim Ali**	**Kacper Bak**
User Study	√	√
Report	√	√

Table 2. Project Tasks

REFERENCES

1. Debian User Forums. Is there an automatic .config generator for kernel 2.6.33.4?, 2010. *Accessed: July 27, 2010.*

2. Kernel Trap. Linux: CML2, ESR & The LKML, 2002. *Accessed: July 27, 2010.*

3. Linux-Kernel. Aunt Tillie builds a kernel (was Re: ISA hardware discovery – the elegant solution), 2002. *Accessed: July 27, 2010.*

4. E. S. Raymond. *The CML2 Language: Constraint-based configuration for the Linux kernel and elsewhere.* 2000.

5. S. She, R. Lotufo, T. Berger, A. Wasowski, and K. Czarnecki. Variability model of the linux kernel. In *VaMoS'10*, pages 45–51, 2010.

6. J. Sincero and W. Schröder-Preikschat. The linux kernel configurator as a feature modeling tool. In *ASPL'08*, pages 257–250, 2008.

7. Soft32 Linux User Forums. kernel autoconfig option?, 2007. *Accessed: July 27, 2010.*