Knowledge, Power and free Beer

Andreas Tille

June 16, 2007

Contents

1 Free Beer

1.1 Hard and soft Ware

1.1.1 Hard Ware

In the field of informatics we differentiate between hardware and software. What does these terms mean?

First lets think about the term ware in general. A key feature of a ware - which is something which is selled by a vendor - is its weight. Very often the price of the ware is proportional to this weight. Both parties - vendor and vendee are able to do certain things like biting on the thing or throbbing at it or whatever.

All these methods might be useful to check the quality and thus the value of the ware to buy - or just for fun. In any case the buyer is able to check whether the ware fits his expectations before handing out the money while the seller is able to prove that the features of the thing justifies the price.

The term computer hardware perfectly fits this features. It comprises all of the physical parts of a computer, as distinguished from the software that provides instructions for the hardware to accomplish tasks.

Once the vendee has bought the ware he deserves certain rights on it. He is now the owner of this ware (provided he has paid the full price he and the vendor agreed to). The person who owns something has the pristine right to alter, change or disassemble the thing. If the ware shows some hidden problems, has some faults or gets broken in a certain time the vendor has to guarantee for the promised features. Finally the owner has the right to sale again the ware that was bought some time before.

As a rule duplication of a ware is not worth the effort because the professional manufacturer mostly can do it much cheaper.

1.1.2 Soft Ware

The features of software are quite different. In fact the question comes up whether it really deserves the term “ware”. The main feature of weight is missing completely because defining a weight for software makes absolutely no sense. Moreover the person who is about to buy a certain software is barely able to check it completely as it is possible with any ware. Even if there are some vending models to use a software for a limited time before buying it. But these days software is this complex that the verification and comparison of competing products is sheer impossible (compared to find out which salesman trades fresh fruits at the market place).

But even if the buyer payed the price for the software the relation to the thing which was payed is quite different than for a piece of hardware which was discussed above. This becomes clear when somebody would take the time to read the strange text which is always displayed of the installation routine of proprietary software. There must be a reason why this hard to understand text which sounds very juridical is bothering the user who wants just to install a piece of software on the computer. But most users have learned that it works perfectly to just press the `''Next''`-button while ignoring the text.

This behaviour prevents users from recognising that they do not really own what they payed money for, they just got a license to use it, sometimes applying some additional restrictions like that the software can only be used on this specific computer and will only work if you send in some data of your computer to get a license key. In case of a piece of hardware – say a chair – this would mean you pay for the chair but you are only allowed to use it in a certain room next to a specific table and only at the gable end; for the other side a different chair with a license to use at this place is needed.

Moreover the license of proprietary software does not allow to change anything. Sticking to the chair-example, this would mean the user is not allowed to paint it with a different colour. Last but not least selling the software is often not allowed. The reason is obvious: The user just does not own the software - the money the user spend on it was for a license to use but not to sell.

Besides this completely different relation between the owner of a piece of hardware and the license owner of some software there is a further difference: In principle it would not be hard to duplicate software. That's why this is not only strictly forbidden but also prevented by technical means.

1.1.3 Fresh fish

Around 50 BC at the market place of a small village in northwest Armorica the fishmonger *Unhygienix* is infamous for his rotting products. This often causes trouble with *Fulliautomatix* and other inhabitants of the village. This means selling bad smelling fish at the market place is nearly impossible. Each customer would immediately notice that it is to old.

But what about software?

The customer has no sign about how old the code of proprietary software might be. It has no odour or other signs of age which are easily to recognise. Moreover software usually consist of several parts which are developed at different times. It frequently happens that the major parts are quite old and only new features and user interface is really fresh.

But what is the problem here?

If somebody pays for a license a certain amount of money only a fraction is payed for really new code. A larger part of the money is spend for code which yet was paid in former versions. But former versions should not cost money. Users just got the license to use this code for years. But what are the stinky bits in the whole software bundle?

Lets recapitulate: Users are perfectly willing to pay for the work of the programmers. It is a fair deal to pay for a service like software development. But the deal is only fair in the case that the user is able to verify how much time was spent to produce this piece of software which was not payed before.

1.2 Patent recipes

1.2.1 The progress

The progress bar is something which keeps impatient users happy. It is the way of programs to inform users that some business is going on and give hints how long this business will last. It is no rocket science. If you see a 100 m sprint you can clearly see start and goal and the runners move like a progress bar from the beginning to the end. So a progress bar can also be viewed as a analogue to real live procedures which are just visualised at the computer monitor. It is hard to tell what the technical invention in the progress bar might be so it hardly can be patented.

But there is `Patent No. EP0394160; Applicant(s): IBM (US)` which exactly describes the progress bar.

The next example for a so called "technical invention" might be another illustration of the issue: It seems reasonable to store different versions of a document in that way that you include the date in the filename. Provided that this date-tagged file archive is stored on a network drive which might be accessible to colleagues than this "technique" infringes `Patent No. DE10108564` which is just valid in Europe.

So what is the issue in declaring common sense as technical invention?

Big software companies acquire a large portfolio of patents - reasonable but mostly such strange things as mentioned above. In case they want to use a technique which is patented by another big company the deal is as follows: Allow me to use your technique and I allow you to use my technique for free. So a large amount of patents is kind of an insurance not to become sued by a competitor. This sounds very reasonable from the point of big software companies and is the driving force behind them to register as many as possible patents.

But what about smaller software companies that do not have this kind of insurance. Or even worse what happens to the Free Software programmer who has not even the slightest idea that his idea he had from common sense is patented by someone else?

1.2.2 EU: Council versus Parliament

This talk is about Free Software. It is not about politics. But unfortunately the topic of software patents in Europe is a very political topic. Unfortunately not many people do care about this. So it remains mostly not regarded if representatives of national parliaments vote for different things than the national parliament has instructed them to vote in the Council of Europe. Less people notice if the rules of democracy are broken for the sake of big software companies. But this would be material for one or two separate talks.

So here is some homework for the interested reader. Just ask your favourite search engine for the following questions:

1. What is the difference between the Council and the Parliament of Europe?

2. Which panel brought up the issue of software patents and what is the suspicion who is behind this issue?
3. What do we owe the representatives from Poland in the Council of Europe?
4. If you are not from Poland what role played the representatives from your own country in the Council?
5. Which panel has the chance to stop the efforts in the wrong direction.

If you are able to answer all these questions you can go straight to your representative which you voted for the European Parliament and ask him to vote against software patents.

1.3 Free of charge

1.3.1 Soft ware is not for sale

In the beginning it was stated that software can hardly be regarded as a ware. This has the consequence that it is hard to sell it for a fair price. There are many people who get this idea but came to different conclusions. Common to these conclusion is the fact that they give away their software at no cost. But what are the differences.

Download here The user can download the ready to run executable. For most users this is fine. They want a program which just does a certain job and do not want to pay money. This is kind of "free beer" - software. You go to the restaurant to get drunk without spending money - free beer works perfectly for this purpose.

But is this really freedom? Does this program do what the user expects it to do? Isn't it just a Trojan horse which spies out user data? Can the user be really sure that it is not harmful?

It just happened that people mixed something into the free beer . . .

Do not change The author of a piece of software allows the user to have a look at the code. There might be several conditions under which the user is allowed to look at the code but assume here that having a view on the code and using the program is free of charge.

This at least might ensure the user that the code does not contain any harmful bits which is not the case in the *free beer* flavour of free of charge software.

Free Software Free Software is according to its definition not only free of charge, it also allows to use the code in complete freedom which means the user is allowed to change the code for his own purpose and can even give away the changed version.

If there is such a lot of software free of charge the world must be a better place. So a normal user might wonder: Where's the catch?

1.3.2 The catch: Free $\neq$ free of charge

The term “Free Software” does not say anything about the money a user has to spend to solve a certain task with this very peace of software. The crucial thing is just that the code of the software can be used without any restrictions by anybody. The fact that the code can be used by anybody free of charge does not implicitely mean that it is really useful to solve a certain task immediately. Any piece of software needs adaptations, maintainance and service. This does not come for free.

The *free* in “Free Software” is more in the sense as “free speech” instead of free of charge. Free speech requires time to write out. It needs competence about the topic and sometimes free speech requires courage. All these are values which can not be countervailed with money – but they are certainly worth anything. So there is either some cost of time for the user in person or the user has to pay anybody to do the job.

The exact definition of Free Software is given in the *Debian Free Software Guidelines*.

2 Knowledge

2.1 Alternatives

2.1.1 WikiPedia

If there is anything in the world which is easily comparable to the fascination of Free Software by sharing many ideas in common than it is free knowledge. The goal to collect the whole knowledge of mankind for the whole mankind for free is the goal of *WikiPedia* for short.

If you are an expert in any field or think you know something special just test the quality of this community effort driven encyclopedia: Go to *wikipedia.org* and look up this special field. There are two basic options:

1. The contents is well written and is right according to your knowledge. Just be happy that you found a new information source and continue checking the contents of other fields if they represent your own knowledge.

2. The contents is not well written, is not precise or even completely wrong - you just are not happy with the contents. There might be people who just surf away and use other sources of information while ignoring this WikiPedia rubbish.

   But once you proceeded through this paper up to this point you hopefully behave differently. You learned something. You just recognised the fact that nothing comes for free. Take a little bit of your spare time and bring your knowledge in precise and well formed sentences and just make WikiPedia better. Your fellow WikiPedia writers will be happy about this and other readers will be even more happy from now on once they stumble over the page which was not satisfying before your editing.

```
Subject: Wikipedia knows the shit
Date: Sun, 27 Mar 2005 16:18:36 -0200
From: Christian <VPYLGHQ@interacti.net>
To: aids-std@rki.de

Newest penis enlargement system:
- Increases the penis length and girth
- Medically Proven (source: wikipedia)
- No Surgery
- Permanent Results
- Proven Traction Method
- 100% Satisfaction Guaranteed

Find more info here: ...
```

Figure 1: Spam mail referring WikiPedia

2.1.2 WikiPedia is taken honestly

Figure 1 shows a quite interesting spam mail. Well, not the contents itself is interesting, but the fact that spammers do refer WikiPedia to “prove” that their rubbish is worth anything. This is kind of: If your enemy takes you honest you must be somebody.

On the other hand it reveals the most common biased opinion about WikiPedia: If *anybody* can write and change articles how can be assured that nobody writes something which is wrong on purpose. (This effect is called “vandalism” in the WikiPedia slang.) The answer is: Many people are watching.

There is a log of every single change of each single page. Reverting some change is done quite quickly and thus the effort of vandalism is high compared to the effort which would be needed to keep WikiPedia clean.

Assume the dirty spammer which sended the spam mail which is displayed in figure 1 had edited WikiPedia first before sending out the strange mail. Even if this person would have done this – it is not in WikiPedia any more. This is one example that the auditing process of the WikiPedia activists is working effectively and even if there is vandalism it does not seem to have a major influenze on the whole collection of free knowledge.

Last but not least there is some solution for the boring spam problem: In figure 2 the output of this spam filter showes that it catched the beast – a nice example for a useful peace of Free Software.

```
SpamAssassin:      17.1    points,    5.0    required
 pts  rule name   description
 0.1  SATIS_GUAR  BODY: Mail guarantees satisfaction
  17  BAYES_99    BODY: Bayesian spam probability is 99 to 100%
```

Figure 2: SpamAssassin handled spam mail

2.2 Free Software in general

2.2.1 As You Like it

What is the basic need of users of any piece of software? What do users really want to solve a certain problem?

A steady and continuous functionality over long period is the main demand of software users. The program has to be safe against failures to not interrupt the work. It has to have a certain set of functions which are necessary to solve a certain task and it has to provide this functionality for a long period in which the task in question has to be solved. Any upgrades of the software do require new learning which just takes time which is cut of from the time span which is reserved for the main task of the user. It is a common missconception that users always want the latest and greatest version of any software project. A further missconception is to overload programs with more and more functions: The more functions a program has the harder ist will be for the user to recognise which functions are really helpful to solve the task in question.

If these requirements are fullfilled the user does not really matter about the concrete program which is finally choosen to solve a certain task. A user simply wants to solve a task and does not want to bother about the philosophy of the tool that is used for it. Just ask your local carpenter whether he cares about the wood which was used to make a handle for the sledge he is using. Most carpenters will not care whether this piece of wood comes from a forest which should be saved or from anywhere else. So users just want a functionality.

In the case of software functionality is not just the program. It is more than that: it is the program and the service which is provided for it. Users have to decide how much money they want to spend for the given task - usually there is some budget for any task. So if there is a given budget and the program itself comes for free there is more money left for the service. Usually service is quite expensive these days so it makes sense to spend more on this side of the calcualtion.

So far to the advantages from the users point of view. But what about the producer of a piece of software? What would a software company loose if they would release their code which proprietary software companies keep as a secret. There are several reasons which speak for releasing the code as Free Software. It is just the problem that everybody thinks it is impossible because selling proprietary binary code is kind of normal these days. But it is really the most effective form to to make money in software business?

At first there are software companies which make all their money based on Free Software like *RedHat*. At second there are companies that see their future in releasing at least parts of their code as Free Software. Well known examples are *Novell* or even *IBM*. The rationale behind this is quite simple: The proprietary software market is currently characterised by a hard competition. If a company recognises that it is impossible to beat the competitor it tries to drift to a new field. In several cases this leaded to a release of code as Free Software (Mozilla, OpenOffice, Interbase, MaxDB, E-Directory, . . .) and providing service for this software (see above for the relation between functionality service and program code).

As a matter of fact providing services for Free Software is simply cheaper than if the code is not available. It is quite common that there are always some users of a certain piece of software that are keen on trying to solve a problem they have themselves. In case they have understand who Free Software works – sharing ideas, work together, join forces – they provide solutions or fix bugs in the software for free. They just have learned that if they provide their work to the authors the next version will contain their code and will make their own work easier because the program has an enhanced functionality. And this is the real clue: The authors of a piece of Free Software get freelancer for no extra cost. The driving force behind these freelancers who volunteer to provide enhancements is that they need a certain functionality to solve a task. They want to spend this time only once and thus they submit their work to the authors. If the authors are part of a company which provide services for this software in this way they got some freelancers at no cost. Compared to external freelancers for proprietary code which requires hiring educated programmers for a lot of money, signing discretion contracts while being unsure whether they do the same for a competitor etc. the Free Software approach has an extraordinary financial advantage for the authors of the code.

The crux hereby is that you need a wide user base because only a small amount of users will really provide reasonable input. On the one hand this is the case just because in most cases users will not be gifted programmers on the other hand users might simply have not understand the principle of Free Software. They regard software as a ready product which just has to be accepted with all flaws it might have and with all problems it might cause when trying to do the tasks a user want to do. To solve the latter problem texts like this one are written.

A wide user base to attract gifted programmers is normally no problem in case of pieces general software which is needed by every user. So finding programmers for an operating system kernel (like Linux), a web browser (like Mozilla), an office suite (like OpenOffice) etc. works perfectly. The problem changes in case of specialised software which is naturally used by only specific user like people working in certain professions.

2.2.2 Get the facts

When the main global player in software business recognised that there is something else which is penetrating the software market. After a certain time they started a campaign title: “Get the facts”. They paid for doing studies which should prove that proprietary software is better than Free Software. No matter what the result of these studies might have been it shows one important thing: They would not pay for studies against somebody who is not taken honest.

BTW, it turned out that a certain amount of the facts was quite questionable. . .

3 Power

3.1 Distribution

3.1.1 Linux from scratch

The usual way a user obtains a software bundle is when buying a piece of hardware. This makes perfectly sense because computer hardware is completely useless without software. Usually a computer comes with a proprietary operating system and some additional proprietary programs to solve simple office tasks.

Assumed a user have heard about the existence of Free Software and decided to try it on the new computer. The reasons might vary from the intent to save money (and buying a computer without any software) to just being not happy with the software which would usually come bundled with a computer.

The first thing which is needed is a Kernel of an operating system. There are more than one option but Linux is the most popular choice. In principle the way to obtain a piece of Free Software is to go to the web page where the source is provided, find a compiler which is able to compile this source and install the program that was builded on the machine. It will take a certain amount of time to install the Linux Kernel and some basic tools which make the “GNU/Linux” operating system. This is described in the *Linux from scratch* project.

The modular nature of Free Software implicitly means that at this point the user does not have a shiny colourful desktop but just a command line interface. The next step would be to get the source and compile the X Window System which is not really a simple task for a beginner. If our user managed the work up to this point he reaches a crossroad: The religion of the desktop environment. The main roads are marked with the signposts “Gnome” and “KDE” but these are not the only ones. The user fails to find a reasonable decision and flips a coin which road to follow. From this point on the screen of the computer looks nice – but there is no program which enables the user to do productive work.

Our brave user now has heard about two shiny new projects: Firefox and Thunderbird. He finally manages to get these programs installed on the computer and can do the first tasks: Browsing the web and managing e-mails. Even if this owner of the new computer started very optimistic with the intent to install Free Software recognises that it was quite a large amount of work to reach this step.

Once it comes to more complex user applications like a comfortable office suite like Openoffice.org, a professional type setting system like LaTeX or a feature rich image manipulating program like Gimp the user is really tired.

If it finally comes to some server software like a web server and a database server the owner of the computer is completely tired. So many decisions to choose from, so many web sites to go - what is the sense to spend so much time?

3.1.2 Distributions on the market

The answer is: Nobody really wants to spend so much time just to exercise the possibility of Free Software: Do it yourself from the source to the installation for every single piece of software. It is not only time consuming it is even error prone to build a complete system. It just should be done by experts.

A complete system containing all the bits of Free Software mentioned above and much more is called *distribution*. It is builded by experts that know the code, use sophisticated tools to build the software and prepare it to be installed quickly onto users machine. This preparation is done in so called *packages* that contain compiled code of the Free Software projects and some preconfiguration to work together with other components of the whole system. Building a distribution from Free Software is kind of a service. Users may decide whether they want to spend days for doing it all on their own or just pay a certain amount of money to buy a distribution. Most users decide it is worth spending some money for a distribution.

But this uncovers some misunderstanding in the relation of Free Software and money: While (at least most) bits of the distribution are free of charge the whole thing costs a certain amount of money. This is one flavour of the "a thing is more than the sum of its parts" principle. The price which is payed is for the service inside and users pay what they regard worth the money. They just have to sum up the amount of money they could gain with productive work in the same time when doing all things on their own. As a side note it should be mentioned that there are also distributions available for no charge but this is explained in detail below.

It has made clear now why users go to a software store and buy a distribution of Free Software. They buy a certain box which has printed in bold letters: "**XYZ** Linux version **A.B**" where **XYZ** stands for the company that compiled the distribution and **A.B** for the version number the distributor has chosen to mark this release state of the whole bundle of software. New users tend to ignore the fact that this is by no means the version number of Linux which is just the kernel of the operating system and currently has a version number below 3. So if somebody is telling you that he is using "Linux version 8.x or 9.z" he has apparently a misconception about what Linux is and what he really bought in the software store. To express this more clearly Free Software enthusiasts speak from "GNU/Linux" because the complete system is more than just the Linux kernel but the many other Free Software tools around which finally make an operating system.

But once we come to colourful boxes with bold letters on it we are back to the commercial world of market place with pressure of competition and gaining market share. It is a known fact that selling GNU/Linux distribution is a normal business where all the rules apply that are valid for vendors of proprietary software. If a distribution company is not able to make profit it will not be able to survive. The good news is that there is a certain number of companies that survive perfectly since several years. Why is this good news? It just proves that it is perfectly doable to run a business on top of Free Software - which is free of charge in its pieces.

But how to compete with other distributors that are just compiling the same basic set of Free Software applications. There are several distinctions between distributors. One is the local aspect. So if a user from Northern America speaks about his new Linux 5 (see above) he is talking about his copy of RedHat Linux while a user in Germany has not really a more recent version of the Linux Kernel if he is talking about his new Linux 10 because he is referring to his copy of SuSE Linux which has just a different version number scheme. People in France prefer Mandrake Linux and Turbo Linux is the main player in Asia – except for China where RedFlag Linux seems to be used mostly.

To make the jungle of distributions even more impenetrably there are distributions that target more to the general desktop user with the latest office applications and providing interfaces to proprietary applications which normally run on other operating systems using emulators (like Wine or DosEmu). Users are free to pick the distribution that fits their needs best. They have a real choice which they do not really have in the case of proprietary systems because there is basically one for a given hardware platform.

Users have a chance to identify themselves with their distribution of choice and who ever have heard over a flaming discussion between users of different distributions knows what competition means.

3.1.3 The missing link

As it was explained in section 2.2.1 Free Software works most effective if there is a wide user base. That's why also distributors decide to include these Free Software projects that are potentially used by many users. From a distributors point of view this makes perfectly sense because serving a mass market implies the distribution will be bought potentially by a high number of people which keeps the business running. It would be wasteful for the distributor to employ specialists for preparing specialised software of certain fields to sparse specialists the work to find out which software fits their needs and how to install this software.

For a moment this sounds fair enough to build a solid base of often needed applications and leave the remaining things for those people who really need it. But the problem is that because applications of special fields do not have a wide user base and thus did not developed so far that failure prove installation methods and easy updating mechanisms are well developed which enable people who are no computer experts to easily proceed with the task to install this software on their computer. The problem becomes worse if the applications require a complex database or web server setup.

One solution would be that small service providing companies take over this job. It is quite common that the authors of a certain piece of specialised Free Software provide their code for free and sell the service of installation, adaptation, documentation and maintenance.

The other solution is that a group of experts builds a complete distribution by just moving the Free Software development principle to distribution level. In fact this is done and it got the name Debian.

3.2 Debian - a distribution builded by a community

3.2.1 Maintainers are experts

In contrast to the GNU/Linux distributions mentioned above Debian is not a company but an organisation. The goal of this organisation is not to sell the distribution. Debian sells nothing. It just produces a free distribution. The people behind Debian who are united in this organisation have just a common goal: To build the best operating system they can afford. While this goal sounds simple the motivation for doing this seems to be missing. While commercial distributors try to earn money Debian people seem to work just for fun.

Well, sometimes it is real fun but there are stronger reasons for the Debian people to work on their common goal: They just get what they need for their own work. They know that they can trust their system and users appreciate their work. The best appreciation users could express is by just providing informations about problems they see, suggesting solutions for these problems or even sending in code which just has to be applied to make Debian better.

But how does this particularly work? The developers of Debian are volunteers who have a special interest in at least one special piece of Free Software. A developer becomes a so called maintainer of a package by just doing the usual work all distributors are doing: Obtain the source of the software from the download area of the software, compile the software and build packages which are easy to install. This is so far no difference to other distributors (note: the packaging format might be different but this is really a technical matter). The difference here is that this maintainer becomes on the one hand the first user of this package – so he is interested personally – and he is the first person to ask if users would run into trouble with this package. It becomes clear that the maintainer is kind of the missing link between the authors of a Free Software project and the final user.

The maintainer is probably one of the most qualified person for this special application inside Debian - and the users know his name. For complex applications usually there is even a group of maintainers who care for larger projects. But each of them knows about the problems that the application might cause in the installation process and tries to solve these with inside scripts and configuration files that reduce the work for the final user to a minimum.

3.2.2 Special applications

But what about the special applications issue which was not solved by commercial distributors who are only able to provide a quite general system? This problem is solved inside Debian because also Debian maintainers work in special fields and need special applications. So if a maintainer works in the field of medicine he will care for the medical applications he needs. Teachers can become Debian maintainers and will probably care for applications that can be used in their schools for education etc. In Debian this principle is called *"DoOcracy"* = the doer decides.

So it turned out that Debian became the largest collection of ready to run Free Software in the internet, containing not only general applications but also programs which only specialists need. They are provided in a way that users can easily install them on their computer. Users of special applications will not be troubled with extra efforts like compiling the source themselves and follow a complicated install procedure.

3.2.3 Lost in the jungle of applications

The largest collection of ready to run Free Software – this sounds great for marketing experts and makes a quite good statistics. But what would this mean for the unexperienced user who just started using Debian and finished the first steps of the Debian installer? Should this poor user start reading the descriptions of 18,000 packages where at best 50% can be understand by a newcomer?

If this would really be the case the applications that are really interesting for the user would be hidden perfectly and it would be nearly the same as if they would be not packaged at all. But there is a light at the horizon.

3.2.4 Custom Debian Distributions

Custom Debian Distributions are completely integrated into Debian. It is a technique which enables a Debian user to focus on these parts of Debian which are really interesting to solve a special task. It is implemented in so called "meta packages" which are simply spoken packages which can only be installed if a set of other packages is installed at the same time. The packaging mechanisms inside Debian care for everything if the user tries to install a meta package.

So in case a user is a teacher and wants to install a computer lab with educational software for the students, focussing to the *Debian-Edu (SkoleLinux)* Custom Debian Distribution is the best idea. In this special case the situation has developed that far that there is even a special CD to install Debian-Edu also called SkoleLinux and the whole Debian system is not really needed, but in principle everything which is on the SkoleLinux CD is included in Debian.

Another example is *Debian-Med*. It was prepared for people working in the field of health care. There are a lot of applications that are used for DNA or protein sequences inside Debian. All of these will be installed when one single package called *med-bio* is installed. The same procedure for medical imaging applications: The installation of *med-imaging* causes the installation of all packages that are relevant for users who have to deal with medical imaging tasks.

This technique dispenses users that are working in special fields from the task to do researches which applications do exist and which are relevant for their own task. The packages will not only be installed at this computer but they also get a menu entry which is made visible in a special user sub menu for easy access to the programs in question.

Conclusion

In the first it was explained why Free Software exists and that "free" in this sense does not really mean "free of charge" as free beer. In the second part also free knowledge (WikiPedia) was explained and things people should know when dealing with Free Software and their relation to proprietary software were mentioned. Finally the third part explained which power is behind Free Software if it is used sanely in a distribution which fits the needs of users.