Syllable

Syllable News #01 August 2007

Inside this magazine:

Welcome to the new SDN!

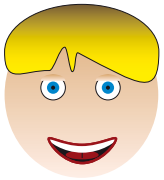
With a new format and fresh content, SDN is the first web magazine for Syllable.
Whatever your level of expertise, we hope you'll find something interesting to read.
SDN will cover news about Syllable and events when they happen.
There is information for users old and new, and articles for developers of all experience levels.
You'll find interviews with developers and contributors from the Syllable community.
Maybe there will be the odd exclusive...
Just like Syllable, SDN is a community effort. If there is something you would like to see in SDN, you can submit your own articles for publication.

Thanks to everyone who has contributed to this first edition of SDN. We hope you enjoy reading!

Kristian Van Der Vliet

Multi-threaded C++

The phrase "multi-threaded C++" can be enough to make even a seasoned developer break out in a cold sweat. Hopefully it won't surprise you to learn that things aren't that complicated with Syllable. The API is written in C++, and it is multi-threaded, but Syllable has a third component that makes things easy: Messages.



By Kristian Van Der Vliet

Understanding how Syllable uses messages is the key to developing with Syllable. Using libsyllable is very different to older APIs such as Win32, or the Slots/Signals system favoured by APIs such as Qt. Libsyllable is most similar to the BeOS API, so if you've ever programmed on BeOS you'll feel right at home on Syllable.

The basic concept of a message is very simple. The Message class is a container for key/value pairs, with a code that acts as a unique identifier. Messages can be sent between applications, or generated by Controls when the user manipulates the GUI. Applications communicate with the various servers, such as the appserver or registrar, using messages. You can use Messages to send almost any type of data you would like.

The most common form is a Message which has no key/value pairs. It is only a unique code that is used to indicate that "something happened", for example the user has selected an item from a drop-down menu. Slightly more complex forms have one or more key/value pairs which provide extra information or data to the receiving application. In its most complex form a Message may contain other messages, or multiple key/value pairs with the same key. This is possible because keys also have an optional index. Don't worry too much about this right now: these types of Message are very rare.

If someone sends a Message, your application must have some way to receive it. There are two classes, Looper and Handler, which work closely together to accept and process Messages. Each of the classes are really very simple. The Looper runs in its own thread and waits for a new Message to arrive by calling the `get_msg_x()` function. When a Message arrives, the Looper passes it to a Handler, which in turn acts on the Message which has just been received.

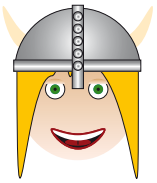
There is a certain amount of artistic licence in the previous description where a lot of the finer details have been glossed over, but from the point of view of an application developer it doesn't actually matter. You only need to be aware that Messages are sent and subsequently received by a Handler, which may be running in its own thread thanks to a Looper.

Sending Messages between threads is what makes multi-threaded C++ so painless. There is no need for multiple threads to share state between themselves. If there is no shared data, there is very little need for any locking. Each thread can be fully autonomous, sending and receiving Messages between themselves instead of attempting to juggle shared variables and avoiding deadlocks. Once you've tried it, you'll never let anyone tell you multi-threaded C++ is hard again!

Now you've got a proper grounding in the theory, we'll start to put it all together in the next article by building a real application.

Local & National Groups

Today, the groups are small, and mostly one-man projects, where nothing happens. I think this is because of the limited number of users, and the fact that the current users are followers and developers, more than they are real end-users. They are just not that interested in national communities.



By Flemming H. Sørensen

Reaching the user

What is it a local/national group can do, that an international community can't do? First of all, they can provide news, and general information in the users' native language. Not everybody is comfortable with the English language, and when we are talking about something as basic as the software that runs your computer, you need to get the information in your own language. It's just like going to your doctor.

This doesn't just apply to news and general information. The most important thing of all, is how we help people. It's not just about the language, and surely the national groups can do that part better, because they speak the same language as the users, but it's also a matter of culture. This is, in end-user relations, the most important asset of the local/national groups.

Because of this, you might see groups, that covers several countries, with similar languages and culture. At the same time you might, in other parts of the world, see several groups with the same language, in the same country, but based on different culture.

Promotion & Media-contact

But that's not all, these groups can do. Far from. They know their local area better, and that way they can promote Syllable far better than a large international group can do. They know the press, and (again) the culture, so they can focus on what is important in that area. When you promote an operating system, do you promote it on its stability? Its price? Do you promote it on the fact that a local developer is working on it? Or something else?

We might like to think it's the first, but most likely it won't be, because we're not talking to tech-people. We're talking to real human beings. People who got bills to pay. People who like to feel they are using "their own" system, through a local developer. Humans are emotional, not logical, and there's nothing wrong with that, we just have to keep it in mind.

Documentation

Documentation is also an area where the groups can make a real difference. Making documentation is not just a matter of writing it once, and then keep translating it, till the day you've got all languages covered. The most obvious thing might be the illustrations, that also needs to be adapted to the right language, but there's also a lot of hidden things, that we should not forget. What a user expects from the documentation might vary from culture, to culture. A German user would not expect the same as a Korean user, because they live in different worlds, and relate to things differently.

Software translation

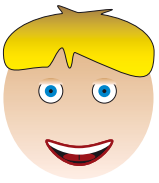
Something else the groups can do, is organizing the translations, so instead of a central place with contact to each translator (the way it is now), it will be the national groups that makes the translations, and send it to the central place. That will also ensure a higher quality, because the groups can provide peer-review, in a way that will be hard to do otherwise, when the number of languages grows.

After having written one page on the subject, without having done anything but touching it briefly, I think it's clear that the local/national groups will play an important role, in the future of Syllable, and is something we should give priority to, and help build. It is my intention to touch that subject too, in a later article.

Collecting garbage

With this article I will try to explain briefly what use you can make of the Boehm Garbage Collector, recently ported to Syllable.

Garbage collection automates the task of keeping track of, and freeing, allocated memory. This makes life easier for the programmer and reduces the risk of memory leaks. Languages like C# and Java rely on garbage collection, while C and C++ require the programmer to manage allocated resources. With the Boehm GC, you can have this feature in C and C++ as well. You can also use it to check your programs for memory leaks.



By Henrik Isaksson

Example 1, checking for leaks:

```
#include <gc/leak_detector.h>

int main(void) {
   char *pointers[2];
   GC_find_leak = 1;
   GC_INIT();
   pointers[0] = malloc(120);
   pointers[1] = malloc(120);
   pointers[0] = malloc(120);
   CHECK_LEAKS();
   free(pointers[0]);
   free(pointers[1]);
   CHECK_LEAKS();
  }
```

The first use of CHECK_LEAKS() will detect a leak; since pointers[0] has been overwritten, you can no longer access or free the memory allocated.

Before you can get started, you'll need to download, compile and install the garbage collector. This is really easy thanks to the wonderful Builder system. Here are the commands you need:

```
build update
```

This will make sure the build system is up to date

```
build get gc-7.0
```

Downloads the GC sources

```
build gc-7.0
```

Configure and build the library

```
build install gc-7.0
```

Yep, this will install it :)

Note: If you get an error saying "unknown thread package" when trying to build, then do this:

```
cd gc-7.0
autoconf -o configure configure.ac
cd ..
build gc-7.0
```

To compile a C program, type

```
gcc myprogram.c -o myprogram -lgc
```

Example 2, collecting garbage:

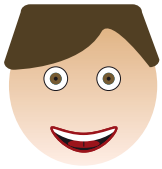
```
#include <gc/leak_detector.h>

int main(void) {
  char *pointers[2];
  GC_INIT();
  {
    char *short_lived_mem = GC_malloc(123);
    // do something
  }
  pointers[0] = GC_malloc(1024);
  pointers[1] = GC_malloc(2048);
  // Do stuff with the memory
 }
```

Notice that there is no free() call here. All memory will be released when the application exits. Memory that is not used may be freed earlier. The "short_lived_mem" in this example may be collected as soon as it is out of scope and no longer accessible. It will probably be held longer, since it's just a single small object, but the point is that you don't need to care about it. The collector will stay out of the way most of the time, and only begin collecting when it's really necessary.

Syllable and Linux

"Why Syllable? Why another OS? Why not just help to make Linux better?" If I had a penny for every time I've seen those questions, I'd have enough for a Big Mac now. Yet this seems to be the prevailing attitude amongst certain vocal parts of the open source community - but it's not just Syllable that receives such flak. Haiku, ReactOS and even the BSDs often suffer the same reaction whenever they get some press.



By Mike Saunders

I'm a Linux fan (and work for a Linux magazine), but I strongly believe we shouldn't invest all our effort into one particular OS. Imagine if Linux was the only thing being worked on, and, somehow, SCO managed to win its court cases. Where would we be? Up a certain creek with no paddles in sight - no projects to fall back on. But thanks to the variety of open source operating systems, if Linux had gotten tangled up in a legal mess, we had FreeBSD as a near drop-in replacement for it.

But but but...

Some of the arguments that people use against Syllable could have been levelled at Linux in its early days. "But it doesn't support all hardware and doesn't have many programs!" Well, nor did Linux in its early days. That's no reason to give up now. "Just give it a few more years and Linux will be on mainstream desktops!" Sorry, but we've been hearing this for nearly a decade now.

Now I love Linux, I love the vast playground of cool technology that it brings in, and it has certainly got a lot slicker in recent years. Without a shadow of a doubt, it's ready for enterprise workstations and many home-user desktops. But there are certain aspects of the OS, such as the lack of a standard desktop, package installation system and set of graphical configuration tools that can't be sorted out any time soon. Sure, variety and choice is good, but for a sizeable chunk of the home user market, these things are definite problems.

When I started playing around with Syllable back in 2003, it had plenty of rough edges. The underlying code was impressive, but the GUI and apps screamed "hobbyist project". Since then, we've seen amazing changes. The desktop has been completely redesigned, many new drivers have been included, and we have a great email client and Webkit-based web browser in the works. One thing has remained the same, though: the focus on simplicity.

Less is more

Much as I applaud the end-user-focused work in recent Linux distributions - and they certainly manage to shield users from the underlying complexity - it's still a tremendously complex OS under the hood. In comparison, Syllable doesn't need as many opaque layers to insulate the user, because it's much cleaner in some respects. Synaptic in Linux may cover up the fact that it's scattering a zillion files all over your hard drive, but there's no need to hide this in Syllable: you just extract a progam into Applications.

That's one example, but there are many areas where Syllable exceeds Linux in simplicity - the kind of simplicity that matters enormously to a big segment of the home-user market. I hope the wider open source community starts to give Syllable the credit it deserves, and not throw tantrums because it's not Linux. Both OSes have their plus points; they can coexist peacefully and productively. The mass-market desktop is still waiting for an open source killer OS, so isn't it good that we have more than one to offer?

Archive instead of a CD

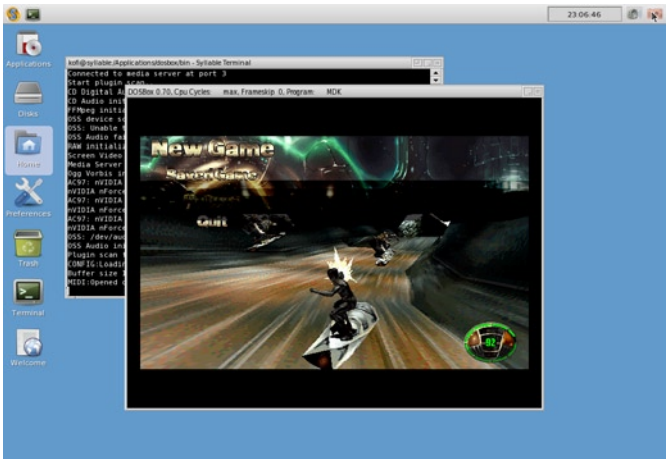
We intended to make a small Magazine CD, but have instead chosen to make an Archive.
In this Archive, you will find a file with a description of the content of the Archive, and how to install it. All software has been tested on a clean install of the latest public release, to make sure it will work for everybody. Some months there might be a lot of software, and other months there might be almost nothing, it all depends on how much software is released.

By Flemming H. Sørensen

In this month we have *DOSBox, Bochs, and Winter.*

DOSBox

DOSBox is a DOS emulator, that will let you run your old DOS programs and games.
The news announcement had this to say:
"The old Syllable port of the DOSBox emulator was never integrated in the build system, so it didn't work any more after SDL was integrated into the base system some time ago. Fortunately, Rui Caridade has ported it again in the newest version. He even got it to run at full speed".



DOSBox

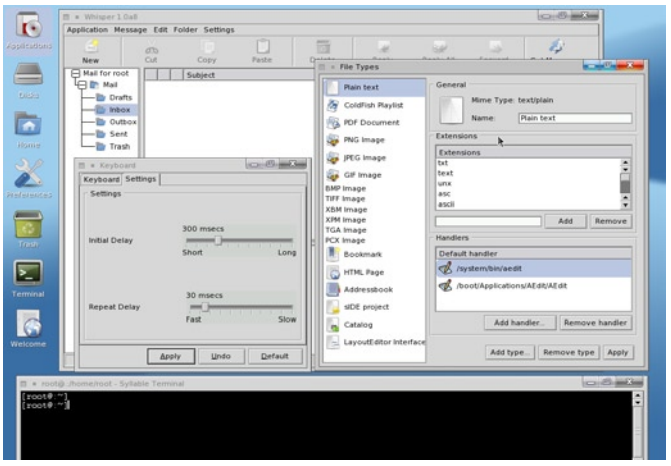
Bochs

Bochs is another emulator, but instead of emulating an operating system, it emulates the whole computer.
The news announcement said this:
"There hasn't been a working package of the Bochs emulator available for Syllable for a long time, but there is now a new version in our downloads. If you are new to Bochs, please refer to the Bochs site for further instructions".

Bochs

Winter

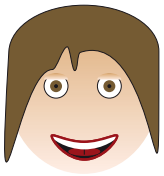
In this first edition of the new SDN, we have the pleasure of something as rare as a new decorator. It has been a few years since that last happened. A decorator should be seen instead of talked about, and that might be why the news announcement didn't say much:
"For the first time in a long time, Syllable got a new window decorator again. John Aspras created the Winter decorator, with a very particular design".



Winter

Indepth interview with Kaj de Vos

When you want to know all about the people working on, with and for Syllable, you need to interview them. This is why I start this new item in the revived SDN: an indepth interview. I'm not sure if I'll ask the same questions to all people here, for becoming too repetitive. Sometimes the first is the best: three pages. Read and learn!



Kaj de Vos

By Ruud Kuin

Once again for who really doesn't know: how has Syllable started?

I'm not the best one to ask because I was a bystander, really. I had only recently found AtheOS, the predecessor project, but was dismayed that it already seemed to be going down the tube. Kurt Skauen, AtheOS' creator, had been making headlines for a few years (as I found out when reading up) by consistently releasing substantial work at a fast pace. Eventually, however, real life caught up with him, as it has a habit of doing. He seemed to be going through a series of small setbacks that conspired to cause him to leave work on AtheOS for a while. This didn't need to be the end of it, but a community had built up around AtheOS and people were expecting a lot from Kurt. He, on the other hand, had always maintained that it was a hobby to him. Consequently, he didn't really want to deal with the community aspects of such a far-reaching project. Actually, he did, but always with the understanding that he didn't really care about that part. This had always caused friction between him and the community that he hadn't really asked for, and when he dropped out of the picture for a while, it came to a climax.
It's a weird and actually rather wonderful story. We have a mythical founder from Norway, who singlehandedly forged a complete operating system that was awesome to behold. Common folk surrounded him to bask in his glory and there were many trolls among them. The code smith grew weary of their unwanted attention and their endless chatter. His interest in his creation waned, and he took it upon himself to embark upon a new journey to explore the skies of the world. Nine months he spent, and built himself an air ship. One day he set sail and disappeared into the blue yonder, never to be seen or heard of again.
This is a true tale I tell you. Kurt bought a second-hand Cessna to learn how to fly. The computer on which he developed AtheOS had broken down, but he didn't want to replace it until he had paid for the repairs that he needed to make to his air plane. We heard this story piece by piece, because Kurt disappeared almost completely, with his machines and Internet connection broken down. He tried to get back a few times, but was angered by what people had said in the meantime. Kurt had never wanted AtheOS to fragment, but one fork had already appeared, to put AtheOS on the Linux kernel, no less. After nine months, both Kurt and the community were fed up. Kristian Van Der Vliet concluded that Kurt was not coming back and announced his own version of the project that he had been working on, under the name Syllable. The other project, Cosmoe, was a fork, and its technical foundation was shaky. It shifted focus over time and didn't attract many contributors. The Syllable project was much more straightforward. It continued what AtheOS had started, collecting all the contributions from other people that were floating around, and opening up the development to the community. The rest, as they say, is history. Just remember that somewhere out there, a mysterious pilot is haunting the skies, with an awesome operating system on his name that lives on as he does.

How did you become involved in Syllable?

At the time, in late 2001, I had been trying to use Linux for a few years. Some of those attempts were successful; others were disastrous. It was clear to me that free software must be the future, but it was also becoming painfully clear what a long road ahead there was. I am an independent consultant and developer. My co-workers were deeply immersed in proprietary technology, but I was trying to add free software to the mix in appropriate places. There was a turn of events where I had to implement much more functionality on Linux than I had wanted to. I succeeded in the end, but not before the whole thing had burned me out. I felt bitten by a raging penguin. In that state, burned out but desperately trying to keep on top of technology gone wild, one evening I was reading one of my many Linux magazines. I think it was Linux Magazine, actually, and it contained an article about AtheOS. It jumped out at

me. It sounded so... perfect. It was clear that here was someone with a brilliant grasp of what needed to be done and the will to do it, and he did it with free software. I started researching on the Internet and didn't stop reading for the rest of the night. At the end of that night, I knew it was the project I had been searching for.

So that's how I became involved with AtheOS: it was an overnight decision. :-) It was right before Kurt's hiatus, and when that story unfolded, it felt like my just-found future was already crumbling before me. Going back was not an option, either to the golden cage of Windows or to the arena to be thrown before the wild penguins. I never told this, but I started plotting escape plans for when Kurt would drop the ball. As it turned out, Vanders (Kristian Van Der Vliet) had done the same, and he was probably the only one who could have pulled it off. While I was new to all this, he had experience and standing in the community and succeeded in gathering a group around him. When he came forward with his project, I joined the discussions for the first time and offered my services.

What do you do for Syllable and how does Syllable support your development?

A wonderful thing happened. I had always been a loner, but for the first time in my life I was working with people at and above my own level on a deeply shared goal. The whole project felt like a puzzle falling into place. We were few, but we all had our own specialities and areas to work on. Even though it was a huge undertaking, it was a very soothing feeling that I didn't have to do everything myself and that we could trust each other to do great work.

I created the build system that I had been contemplating. It takes all the program source code for all the parts that make up the system and uses it to build the system, every time a part has changed. Or you can build single applications with it. Such a system makes full use of the open nature of free software, and it is essential to scale up the creation of a large project such as Syllable. It took a long time to reproduce what Kurt had done in AtheOS and make it repeatable. Only in the past year, after five years, did we reach the point where it was fully automated and dependable. By that time, however, all parts of the system were updated and we had reached a level of great flexibility.
Most of my work stems from this main project. I still develop the build system further and update system components and third-party programs regularly. I do many other things all over the project, but they are usually to support this main work. With few developers, everyone has to be a jack of all trades.

What do you think of the last Syllable release; 0.6.4?

The best thing since sliced bread, of course. ;-) Well, it's really a good release. In terms of software building, we have reached the state that you can install a clean Syllable without extra fixes, and the development tools, and that's enough to build the complete system from source, with up-to-date components. On itself, because Syllable has always been what is called self-hosting. In terms of stability and polish, 0.6.4 is the result of two releases that we have produced under formal, public bug tracking, which has shaken out a lot of the smaller bugs that were still in there. Last year we already took care of most of the big bugs. In terms of applications, Syllable just became a lot more usable due to our new web browser. There are also the new address book, improvements to the email program, and a number of ported applications that have been contributed lately.

How do you see the future for Syllable and yourself?

Well, I've had a bright future all my life. :-) Syllable fits in with that. We're going to do great things.
My professional field is groupware, and besides software building, that's what I've been planning to do in Syllable. It has been a long road, though. First we needed to build the basic operating system. We are now getting around to interesting new projects. The Syllable Server that I have been building is the start of that. The informed reader will know that it's built on the Linux kernel. It means that I have gone back into the arena and subdued all the evil penguins! Total domination! They will now forever be my servants. They will do my bidding and run my groupware for me.

What are the main obstacles today in the development of Syllable? Participation, tools, something else?

Participation, and that follows from usability. Since we don't have many applications yet, few people really use Syllable. With no real stake in the project, they don't feel like contributing. That said, a lot more can already be done with Syllable than people realise. So we must make that known, certainly now that we

have reached a higher level of usability, and this is where the Syllable Newsletter can help. But above all, we must make the system more ready for production use. Our ace card to do that is Syllable Server, which will be stable long before Syllable Desktop, and will have applications sooner. In terms of tools, the only thing we really lack is a high-level programming language that can be used to write graphical applications. We plan to fix that with REBOL.

Comparing your project to other hobby OSes, where do you feel that your project excels and where does it lack?

We have always been ahead of other alternative OSes in terms of the hard underlying values of an operating system, such as design, efficient resource use, stability, hardware support and build properties such as self-hosting and third-party application support. This is hard to see if you don't look under the surface or don't know where to look. Other systems have appeared to be more advanced by just polishing their looks or porting popular applications while the underlying system still has a hard time to support them. We do it in the order that is right in technical terms and haven't spent our efforts on promotion much in the past. This is changing, because our base system is now robust enough to entrust it to early users and start putting the polish on.
Getting these basic values right is a lot more difficult than the polish. We expect to be accelerating compared to some of the other projects that are trying to do it the other way around. Apart from this thin veneer, the only thing we lack compared to projects that recreate an old system is an existing community. While this is a difficult starting position, I have been in a dwindling community in the past, and I wouldn't want to be in that situation again. Syllable is looking towards the future, instead of the past.

What kind of apps are in general missing from Syllable that users usually ask for?

Users ask for a lot, so I'm not sure that's a good metric to go by if you don't have a thousand developers. We always work on things that are within reach. A long journey has to be made in small steps. We have most of the applications covered that you would expect to find included with an OS, but in many cases they're not mature yet. Most others are simply missing, but we don't have the time to work on them. We need to attract contributors to do those.

What would you advise other developers out there that might want to write/participate in Syllable?

Start small. Many inexperienced programmers try to do something big and fail. Computer systems are so complex that they can only be mastered in small steps. Break problems down in steps as small as possible. Don't even start by programming, start by installing the system and looking around. This goes for other tasks as well. Don't start a complete new web site, or a complete book, start by contributing small things to the sites that already exist.
The other thing is to focus on producing. Anything. Don't keep mulling over your master design without producing anything in the meanwhile. If you don't produce the first step, and the second, and the third, you will never finish the last step. More importantly, all the steps in your master plan, except maybe the first, are wrong. I am speaking out of experience. You will know, too, when you gain experience, but if you don't produce all the time, you will never gain the experience to know what's wrong with your next step. You will not be able to have a meaningful conversation with the other developers, either, because they have found out that they have to work on steps that you don't see yet.
Take down one penguin at a time. :-)

What technical and other challenges are lying ahead for these devs?

In many cases, psychological problems. As I mentioned, you can imagine that a problem is so big and intimidating that you never start on it. You can try to do things in the way they're customary on other systems and end up in a fight with the system - or with the other developers. :-) Syllable is different enough that you will have to adjust some of your thinking. You may hold some wrong beliefs that people have told you. For example, you may have heard that multi-threading is difficult and therefore evil. Well, it is on some systems, but in Syllable we don't like absolute rules. The goal is not to be perfect, because this will paralyse you and will keep you from producing the next step. The goal is to be on the way to perfection, and to believe that you can come close. Multithreading, for example, is done right in Syllable. It results in huge advantages such as a very responsive user interface and is still easy to program. But you will have to learn how it works. Step by step.

Thanks Kaj!

Let's talk about the users

The users. How do we get *(more)* users? It's not really a fair question, because with this question, we're implying that the users are there for us. We actually compare them to drivers, programs, etc., as if they were a part of the system. Something we can form and change, as we please.

By Flemming H. Sørensen

Users are not there for us, but when looking at most operating systems today, it looks like the developers think so. We are there for the users, and we have to keep that in mind, in everything we do, or we will end like all the other operating systems out there; user-hostile, instead of user-friendly.
If we make a system, where we need to educate the users, we have made it the wrong way. We should not educate the users, we should get educated by the users.

Now, do we have to change the way we talk? No, this is just about consciousness raising. When you say "One man, one vote", you don't mean "One male, one vote", you mean "One person, one vote". It should be the same, when we talk about the users.

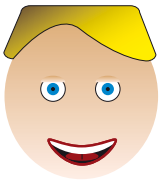
Non-Latin(-script) translations

By Flemming H. Sørensen

Syllable supports more than just Latin-style scripts. Good examples of this, are the Russian, Bulgarian, and Greek translations. A few weeks ago, we even got a Chinese translation of aclock. It works, but it might be worth noticing that we don't support CJK input, so the translation had to be made on a non-Syllable system. Personally, I'm looking forward to a full Chinese translation.

We don't do quite as well, with the right-to-left scripts. We tried an Arabic translation, but it turned out unsuccessful, but one day in the future, support will be added, and it will be possible to have Arabic and Hebrew translations too.

You can help too!



By Ruud Kuin

I hope you enjoyed reading this first version of the **"revived" SDN**. It is our wish to keep you informed on the development of Syllable in all its stages through this e-magazine. If you think you need to share information with us or with newbies, about Syllable or any program used by Syllable, or program you think that needs to be ported for Syllable, feel free to write an article.

This e-magazine is meant for everyone who is busy writing code, testing OS'es or just interested in Syllable, so don't hesitate to write!

Take a look at our site:
www.syllable.org
If you want to share an article with us, mail:
sdn@syllable.org

Glue here

Glue here

Syllable

Live CD version 0.6.4

1. Insert

Insert the Syllable CD-rom in your CD or DVD-drive.

2. Boot on CD

Follow the boot instructions.

3. Be amazed!

Try Syllable without installation!

Syllable