POLYTECH MONTPELLIER

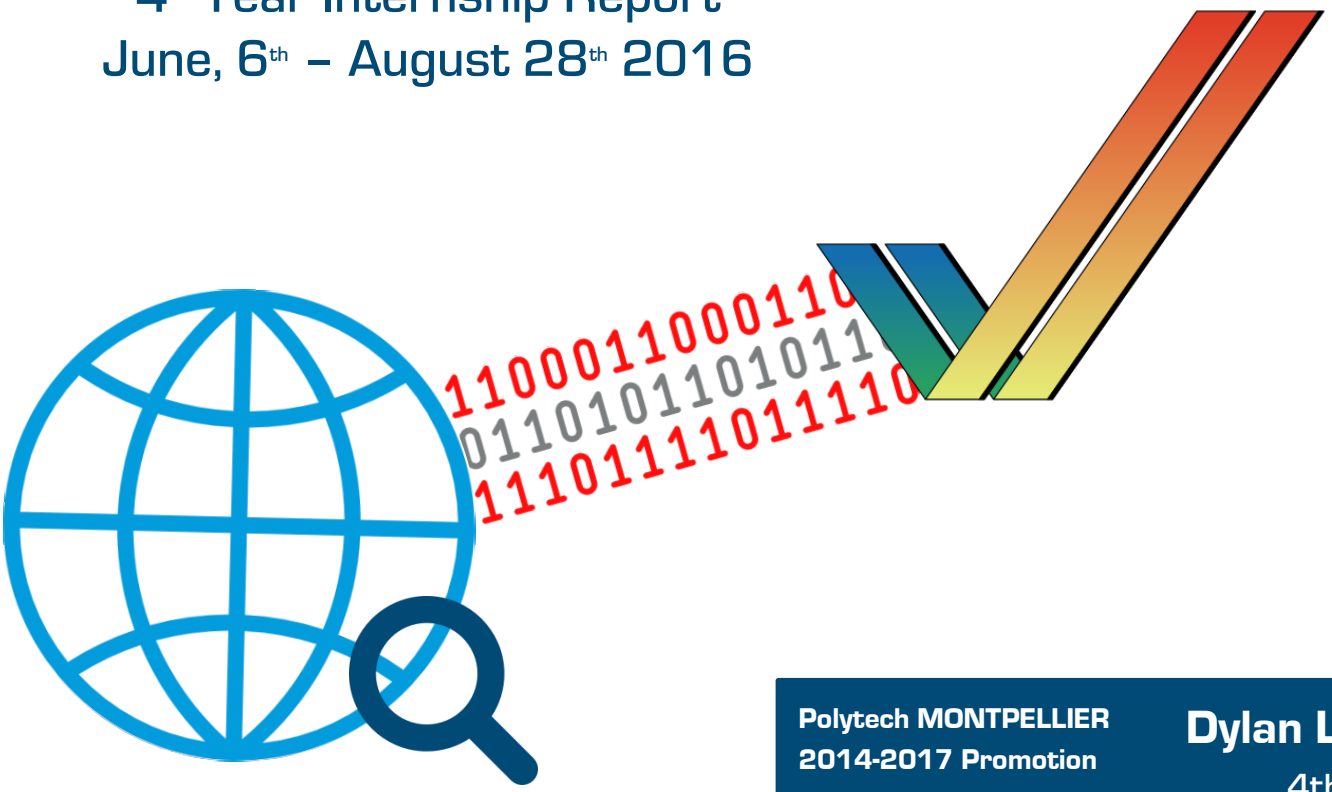
UNIVERSITÉ DE MONTPELLIER

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Université de Montréal

Porting a Web Browser to AmigaOS3

4th Year Internship Report
June, 6th – August 28th 2016



Polytech MONTPELLIER
2014-2017 Promotion

Dylan LEVY
4th Year
Computer Science & Management Dpt.

Internship Tutor
Yann-Gaël GUEHENEUC
Associate Professor
Ptidej Team

IG

Polytech Tutor
Vincent BERRY
Professor, Researcher
LIRMM

Internship done in the Computer Science and Software Engineering Department
at École Polytechnique de Montréal (PolyMtl),
2900 boul. Édouard-Montpetit. Montréal. Postal Code H3T 1J4. Québec. Canada

Content

Figures

*In the report, the words with an asterisk * are explained in the Glossary.*
Only the first word occurrences have an asterisk.

Acknowledgements

I would really like to thank these people and institutions, who/which allow me to perform my internship on this project (the names are given in alphabetical order).

People

- **Mr Vincent BERRY**, my Polytech Montpellier tutor, for his help about this report ;
- **Mr Yann-Gaël GUEHENEUC** aka Tygre, my internship tutor, who gave me the opportunity to work on this old-school but smart project ;
- **Mr Erwan JESTIN** aka JBam, from BPJ Studio, for his AmigaOS3 Raspberry Pi Version called Happiga ;
- **Mr Fábio PETRILLO**, a professor from PolyMTL, for his help to understand better the differences between AmigaOS3 and Windows 95 ;
- **Mr Krzysztof SMIECHOWICZ** aka Deadwood, the Odyssey Web Browser actual developer, for his help to understand Odyssey incompatibilities with AmigaOS3 ;
- **Mr Chris YOUNG** aka Chris, the NetSurf developer responsible for porting NetSurf to Amiga systems, for his precious help during the project ;
- **Mr Aymen ZALILA** aka EyMenZ, my project predecessor, for his help to understand the file containing his guidelines to compile NetSurf for AmigaOS3; this file inspired me to create NetScript.

Institutions

- **Amiga.org forums** (and their users), for their support about NetSurf and NetScript ;
- **Polytech Montpellier** and **École Polytechnique de Montréal** (PolyMTL), which gave me the opportunity to do my internship abroad.

Introduction

In this report, I explain my 3-month internship project and its results as partial fulfillment of my 4th year in Computer Science and Management at Polytech Montpellier: porting a Web Browser to the 3.X versions of the Amiga Operating System (aka AmigaOS 3). Amiga OS is an operating system released in the eighties reputed to be easy to use, efficient, and reliable especially for computers with low resources.

This project took place in the Computer & Software Engineering Department at École Polytechnique de Montréal (aka PolyMTL). This project was based on the work done in winter 2015 by another student from PolyMTL, Aymen ZALILA, who was in charge of looking for a Web browser compatible with AmigaOS3[1]. He chose the NetSurf browser, wrote a file containing guidelines to compile NetSurf for AmigaOS3, and did a first compilation for AmigaOS3.

The problem is that the guidelines were not compatible for the new version of NetSurf, released since then, and it was impossible to compile NetSurf from another computer than the one Aymen used for the first compilation (for example, Yann-Gael GUEHENEUC, my tutor, could not compile NetSurf on his own computer). Moreover, some manual updates were needed on some files. Finally, NetSurf had some bugs, especially one with its JavaScript engine called DukTape. So, it was impossible to use NetSurf with DukTape.

My first role was to fix the guidelines. Then, I had to write a script that automatizes the compilation (inspired from Aymen's guidelines) then make it portable. I also had to track and fix the bugs related to DukTape[2]. Finally, I had to import NetSurf to Happiga, an AmigaOS3 emulator-based distribution for the Raspberry Pi micro-computer (also known as RasPi or RPi) belonging to the Internet of Things (IoT). A part of the work done on this project resulted in NetScript, the script that I created to compile NetSurf for AmigaOS3 easily and automatically on Windows (via Cygwin*)[3].

In this report, at first, we will see the context of the internship and the project in which I was involved. Then, we will travel through the history of the Amiga. Afterwards, I will explain the reasons for importing a browser to AmigaOS3 while AmigaOS4 and other well-known OSes exist. Next, I will focus on the work done to create NetScript. Finally, we will see the work done to debug the DukTape issue and to import NetSurf to Happiga.

---

[1] Aymen's report is available in the Bibliography.

[2] Later in this report, we will mention these bugs as the DukTape Issue.

[3] NetScript is available on the GitHub website.

1 - About the Project

1.1 - Welcome to Ptidej team!

During my three months internship, I was hosted by École Polytechnique de Montréal, in the Ptidej team. PolyMTL is a famous engineering school in Canada, where it is possible to learn different courses in multiple departments such as Computer Engineering, Electrical Engineering, Civil, Geological and Mining Engineering... The school is divided into two buildings, the main original older building, where the majority of the courses are given, and the new building named Lassonde, in reference to Claudette MacKay-Lassonde and Pierre Lassonde, two graduate students from PolyMTL who funded this new sustainable pavilion. This pavilion contains most of the department labs, including the Computer Engineering one, where I worked during my internship.

Figure 1 - The Lassonde Pavilion

Figure 2 - The Computer Engineering Department

The Computer Engineering department contains the department administration, the professors' offices, and the different Computer Engineering labs. Those labs are shared between many different teams. For example the Ptidej team (for Pattern Trace Identification, Detection, and Enhancement in Java), the Soccer team (for SOftware Cost-effective Change and Evolution Research), the SWAT (for SoftWare Analytics and Technologies for the cloud)... During my internship, I met different members of those teams, but I mainly worked in the Ptidej team, managed by Yann-Gaël GUEHENEUC, my Internship Tutor.

The Ptidej team develops different solutions to "understand, evaluate, and improve the quality of software systems by promoting the use of idioms, design patterns, and architectural patterns"[4]. I worked on a future subject related to the team that is important for Yann-Gaël: porting an Internet browser to AmigaOS3.

[4] This quotation comes from the Ptidej website, available in the Bibliography as *Web of the Ptidej Team*.

1.2 - My mission: compile a Web Browser for AmigaOS3

My default mission was to seek for possibly AmigaOS3-compatible browsers, especially some using WebKit (Apple's Framework*) to bring a powerful browser, equivalent to Apple Safari, to an old operating system and thus to shrink the gap between AmigaOS3 and any Linux distribution, in terms of features.

Figure 3 - First step: looking for possibly AmigaOS3 compatible browsers (From LEFT to RIGHT, Timberwolf, Odyssey, NetSurf, Chrome, and Safari)

Figure 4 - Second step: looking for a way to compile the browser for AmigaOS3

Then, I had to compile the selected Web browsers to AmigaOS3, so to make the browser compatible with it. To do this, I had to seek for information on the way the browsers are built. Afterwards, I had to find a compiler or a way to compile the browser for the specific AmigaOS3 architecture.

A previous work on this project was made by a PolyMTL student (Aymen ZALILA), focusing on the NetSurf Browser. Aymen managed to compile NetSurf for AmigaOS3 but the problem was that we needed to do multiple difficult steps (like modifying some files very quickly because the compilation created and deleted them rapidly). Moreover, the compilation was not consistently successful (due to NetSurf recent updates). So, I also had to work on this to fix the different issues and to simplify it.

I had to fix some bugs in NetSurf too. Indeed, NetSurf had a major bug with its JavaScript engine called DukTape. Even if DukTape worked alone in AmigaOS3 and so did NetSurf without DukTape, the combination of both threw an error which was not caught by NetSurf. NetSurf developers did not know from where the issue came. So, a part of my job was to focus on this bug particularly (and on other minor bugs if I had time to).

Later, I proposed to my tutor Yann-Gaël to import NetSurf to Happiga, an AmigaOS3 distribution for Raspberry Pi, as the IoT is a trendy phenomenon and Happiga would be a good platform to test NetSurf. Yann-Gaël liked the idea and accepted to add it to my mission.

2 - AmiStory: The Amiga History

2.1 - A blessing in disguise...

Figure 5 - An Atari 2600 gaming console

In 1980, Atari, the famous American company developing/editing video games and releasing game consoles, was the leader in these domains. Computers and video games just started to be popular all over the world and the Atari 2600 was one of the most famous gaming consoles. This gaming console had quite a complex 8-bits* architecture and design. An 8-bits architecture (simplified in 8-bits) means that, for each data passing through the machine, there are only 256 possibilities (2^8 possibilities).

Jay Miner, an Atari architecture developer, advised Atari to develop a new computer with a new architecture, based on a new processor* from Motorola called the 68000. This processor was more powerful than what could be found on Atari machines, because it had both a 16-bits architecture (to communicate with the rest of the computer parts) and 32-bits architecture (to exchange data inside the processor) which meant 2^{16} and 2^{32} possibilities (respectively 2 times and 4 times better than the Atari 2600 8-bits architecture). This processor was also equipped with a new system called real-time clock, used to trigger some events more precisely, compared to the other processors in the market.

Jay Miner was convinced that using this processor would be a way to outpace competitors. However, Atari thought differently, choosing to rely on their 8-bits architecture processor. Consequently, Jay Miner decided to quit Atari.

A couple of years later, in 1982, Jay Miner joined the new company Hi-Toro thanks to Larry Kaplan (a former Atari colleague). This company, based in Santa Clara (USA), wanted to develop video games and peripherals for game consoles. Larry Kaplan started to develop a computer under the "Lorraine" code name (for Dave Morse's wife), which will become later the first Amiga computer. This computer was designed to simplify video game and software development and to be accessible to third party companies*, unlike Atari which wanted to prevent third party developers developing new games for future Atari computers. Hi-Toro changed its name to Amiga, a friendly name coming alphabetically before Apple and Atari, two competitors.

Figure 6 - The Amiga logo during the 1980's

In 1983 came the video game crash. Atari was on the verge of bankruptcy, trying to create an agreement with Amiga that did not succeed. Atari sued Amiga, which was not in a good position, too. Another competitor, called Commodore, did better than the others, due to the release of the Commodore 64 (or C64) in 1982, a personal computer equipped with 64 kB RAM, which was totally new at this time. Commodore was interested in the Lorraine project and decided to purchase Amiga in August 1983. The two companies merged and became Commodore-Amiga Inc[5].

2.2 - Amiga innovates with AmigaOS and their computers

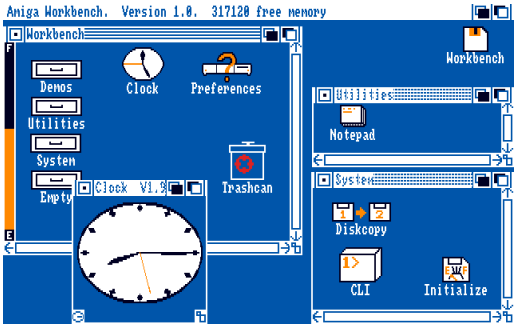


Figure 7 - The Amiga Workbench 1.0 User Interface (September 1985)

In September 1985, the first Amiga computer (the Amiga 1000) was released. It had its own operating system: AmigaOS1. AmigaOS1 had several advantages compared to IBM PC and Apple OSes thanks to its user interface (or UI), called the Workbench. Workbench UI was one of the first updatable UI (AmigaOS1 really means AmigaOS with the Workbench 1.X). It was colorful and intuitive for the time. Indeed, IBM PCs used a text-based OS* while Apple computers were using a black and white UI.

The Amiga 1000 was also versatile technologically. It was possible to connect it to a TV, which was not common then. It was also the first computer to manage to do 3D animations. Moreover, it was equipped with 4 audio channels (the computer was able to play 4 sounds at the same time), which was revolutionary at the time.

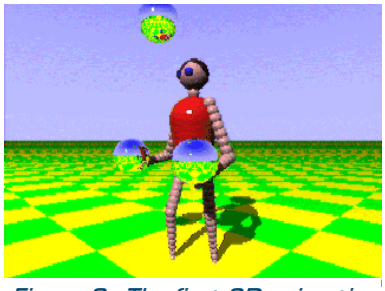


Figure 8 - The first 3D animation available on Amiga 1000: a character juggling (1986)

Figure 9 - An Amiga 500 computer

The one problem of the first Amiga computer was its price, £1,500, whereas Atari had developed and released some months ago a 16-bits game computer called Atari ST at half the price. This release made Commodore rethink its marketing. Commodore developed cheaper computers, accessible to "the masses" (according to Jack Tramiel, Commodore founder): the Amiga 500 and the Amiga 2000. These two computers were sold much cheaper than competitors and became famous, particularly in Europe (6 million units sold for the Amiga 500).

[5] The company was often called Commodore only. Thereafter, I will use "Commodore" to talk about Commodore-Amiga Inc.

Those computers were responsible for the 8-bits to 16-bits transition. Lots of games were available exclusively on Amiga computers, which could explain their success. In fact, the first aim of Amiga computers was to be game consoles only, but thanks to their possibilities, they became real computers. AmigaOS2 was released in 1990 and AmigaOS3, the one on which I worked, in 1992.

2.3 - Too much competitors

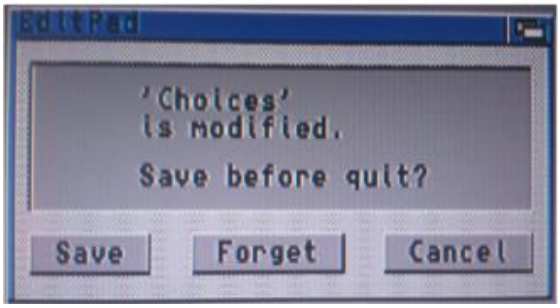


Figure 10 - Saving a file on AmigaOS3
The result is the file is modified in the computer, and either the machine save the modifications or forgot them (the words chosen are important)

Although Amiga computers were popular at the end of the 1990s, Commodore had lots of difficulties. Many serious competitors appeared during this period, such as Microsoft (with Windows 95 and Windows 98), Apple (with MacOS and the iMac computer) and UNIX/Linux OS. Those systems (except UNIX/Linux) were more user-oriented than AmigaOS, so they were easier to understand. Their whole interfaces were designed to give the user the impression of monitoring the system directly, using active interrogative sentences and easy-to-understand answers. AmigaOS, even though it had a user interface, was more machine-oriented, with passive questions and machine-oriented answers. So, Amiga computers (and AmigaOS) became outdated in terms of comprehension and features compared to Windows and Apple. Amiga computers were weaker than the new PowerPC and i86 architectures (more reliable and more powerful). Amiga tried to release new versions for AmigaOS to set things right: AmigaOS 3.5 in 1999 and AmigaOS 3.9 in 2000, but it was too late.

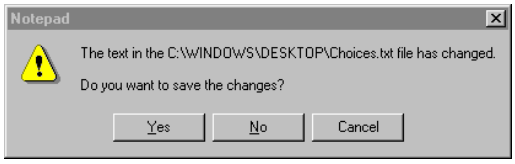


Figure 11 - Saving a file on Windows 95
The file has changed (modified by a user or the computer), and either YOU choose to save the changes or not

As Amiga computers were game consoles first, we could do the same analysis in the gaming world. Commodore released the AmigaCD32 in 1993, a console which could use CD-ROMs (with a better capacity than old tapes and disks). This was the first console with a 32-bits architecture, more powerful than a 16-bits one. Unfortunately, the 16-bits game consoles had already gained dominance and the game consoles market was full (SEGA Megadrive/Genesis, NEO-GEO, Super Nintendo/SNES...).

Figure 12 - The Sony PlayStation game console

Moreover, in 1994, the first Sony game console, the PlayStation, was released and quickly became a phenomenon. As the AmigaCD32, the PlayStation had a 32-bits architecture and could use CD-ROMs, but the PlayStation also had killer games (Tomb Raider, Crash Bandicoot, Spyro, WipeOut, Tekken...), wonderful graphics at the time and a well-thought marketing campaign.

Commodore sank at the end of the 1990s and the beginning of the 2000s (it went bankrupt). The company was bought in pieces by several companies, and it is difficult now to know who owns the Amiga rights.

2.4 - Still alive

Fortunately, Hyperion Entertainment gathered most of Amiga rights and licenses at the beginning of the 2000s and now owns them. The company released a final AmigaOS edition, called AmigaOS4.1 (AmigaOS4 to simplify). AmigaOS4 gave up the m68k architecture and was adapted to the PowerPC architecture. AmigaOS3 software does not work with AmigaOS4, but AmigaOS4 has an m68k emulator, corresponding to the m68k architecture used by AmigaOS3 to run its software.

Figure 13 - AmiKit, an AmigaOS3 emulator for Windows

The Amiga community (often called Amigans) seized the Amiga universe (including AmigaOS) and now tries to enhance it and revive it. Amiga-compatible hardware is still produced by A-Eon Technology and Amigans still use their Amiga computers. Some Amigans use emulators, such as AmiKit (an emulator based on AmigaOS3) or Happiga on the Raspberry Pi (mentioned later in this report). Many Amiga forums on the Internet are very active and deal everyday with remembering Amiga memories, fixing issues or developing software (such as NetSurf). Even if we do not hear about it, the Amiga world is still alive, active, and ready to help anyone who would like to discover this universe.

3 - Why a Web browser for AmigaOS3?

3.1 - AmigaOS3 vs. AmigaOS4

There are currently two main versions of AmigaOS: AmigaOS3 and AmigaOS4. The question is why these two versions coexist? Why people using AmigaOS3 do not move to AmigaOS4?

The first explanation could be the same that people use to explain why they do not move from Windows 7 to Windows 10. Amigans using AmigaOS3 simply love their OS because they use it every day (or often) and they do not want to update themselves because they are worried about losing their bearings. They get used to do what they need with AmigaOS3. People habits is one of the most important reasons people do not update their OS (whether it be Amiga or Windows).

Figure 14 - AmigaOS 4.1 (2008)

The second one is more hardware-specific. AmigaOS3 and AmigaOS4 do not have the same architecture. AmigaOS3 works on an m68k architecture whereas AmigaOS4 uses a PowerPC one. The latter is newest than the first and has more features. Even if it has become obsolete now, PowerPC is still heavily used in industry compared to the old m68k architecture, abandoned during the late 1990s. Moreover, these two architectures are sensibly different. That means m68k-based software are not compatible with PowerPC ones, and vice versa. So, Amigans using AmigaOS3 cannot make their software work on AmigaOS4. To do this, they need to use an emulator included in AmigaOS4, which is not accessible for all Amigans because AmigaOS4 is charged and also because using an emulator can be difficult for novices.

3.2 - Amiga Revival

Amigans would like to share their favorite machines with people who do not know Amiga, but also would like to widen the offering between Mac and Windows. They would like to make Amiga revive again via AmigaOS3, because they consider it as the most known AmigaOS m68k version, with the most features. Also, Amiga computers were first game consoles, so many retro gamers* want to keep it alive and if possible to come it back into fashion. Additionally, AmigaOS can help people to understand better the technological progress done until then.

Moreover, many people continue to develop software for AmigaOS3 (NetSurf is a perfect example) and also organize conventions, such as the AmiWest, in Sacramento, California (October 8th-9th 2016).

On top of that, some Amigans try to spread AmigaOS by adjusting it for new Internet of Things systems, such as Raspberry Pi, with the Happiga distribution.

3.3 - There already are Web browsers!

When I looked for compatible Web browsers for AmigaOS3 (and so m68k architecture), I could see two Web browsers already existed for AmigaOS3: IBrowse and AWeb. So, why should we find another Web browser while AmigaOS already has two?

Figure 15 - AWeb and IBrowse, the two Web browsers already on AmigaOS3

While testing these two browsers[6], I could see that both have a Bookmarks Manager and a search engine (to find a word in page) but not a tab manager. Moreover, they both have some display issues and problems accessing SSL* Encrypted Web pages. Also, IBrowse is only available in a 30-minutes demo version[7], while AWeb became open-source in 2002.

So, there are still some bugs and missing features we can find on every Web browsers currently available on AmigaOS3. What we saw in this part explains a piece of the question "Why a Web browser for AmigaOS3?": we must find compatible Web browsers that will be enough powerful to compete against actual Web browsers.

---

[6] The comparison is available in the Appendix and is named "AmigaOS3 Web Browsers Comparison" (Page 30).

[7] The unlimited version of IBrowse exists, but it is currently unavailable for purchase.

3.4 - Asking Yann-Gaël GUEHENEUC

Figure 16 - Yann-Gaël GUEHENEUC

To obtain more information about the question "Why a Web browser for AmigaOS3?", I asked Yann-Gaël GUEHENEUC, my tutor and the Ptidej Team Leader.

"Operating systems used to be complex but low-level pieces of software, in charge of controlling hardware systems and providing a unified access between different hardware systems and software. Software libraries were developed in parallel to factor out common code and enhance software applications. In the recent years, with the advent of the Windows and Linux operating systems, there has been a tendency for developers of operating systems to include more and more sophisticated libraries in their operating systems to the point where operating systems ship nowadays with APIs to access TCP/IP stack, relational database management systems, 3D graphic layers, and much more. This tendency is also leading different, competing operating systems to get closer and closer. A case in point is Apple operating systems, which is now built around a BSD code base, essentially a Unix, which is (to a certain extent) compatible with Linux APIs. This tendency towards a convergence of APIs brings questions about APIs: what are "good" APIs? What makes APIs successful? How can different operating systems integrate same APIs seamlessly?

Therefore, we are interested in studying the relationships between operating systems and libraries in general and the integration of various APIs in operating systems in general[8]. To study these relationships, we need operating systems, libraries, and software applications that are both simple enough to be studied exhaustively and that are also representative of general operating systems, libraries, and software applications. Consequently, we turn our attention to operating systems with low size footprint, yet excluding operating systems for embedded systems, which are too particular. We choose the AmigaOS operating systems because it is modular, simple, written in C and minimal while not being minimalist. We choose the DukTape library for its simplicity yet usefulness in interpreting JavaScript code in C. Finally, we also choose the NetSurf software application, which is an open-source, multiplatform, modern Web browser."

[8] The source is available in the bibliography and named *[Report] Proposed Research Program, Yann-Gaël GUEHENEUC* (The sources of this report contain articles)

4 - Before NetScript

4.1 - Browsers Pretenders

My first role was to look for Web browsers which could be compatible with AmigaOS3, and so the m68k architecture. In fact, dozens of Web browsers exist: Google Chrome, Apple Safari, Mozilla Firefox, Microsoft Edge … But most of them are developed for more popular architectures (Intel i86, currently the main architecture in computers' world, or ARM for Android phones). So, these pretenders cannot be accepted unless there is a way to compile them for an m68k architecture.

First of all, it is impossible to import closed-source Web browsers, because we do not have access to their source code and doing this would be illegal. So, from the outset, we can exclude Microsoft Web browsers (Internet Explorer and Microsoft Edge).

Then, some browsers are based on a famous and powerful framework called WebKit. Yet, this framework is also based on an i86 architecture and its porting to an m68k one would be complex due to an endianness* issue. With Yann-Gaël, we decided to forget the Web browsers using WebKit. We made this choice because this was too complex but also because this technical prowess would not correspond to the future project Yann-Gaël wants to set up. So, Google Chrome and Apple Safari were excluded (see Figure 3, page 4).

Focusing on AmigaOS4 browsers, Timberwolf, Odyssey and NetSurf could be good pretenders. But Odyssey uses the WebKit framework, and AmigaOS4 works on a different architecture than AmigaOS3. So, we can exclude at first glance those Web browsers.

However, different projects around NetSurf and Odyssey were in progress:

- About NetSurf, a previous student (Aymen ZALILA) worked on guidelines to compile NetSurf for AmigaOS3 (thanks to Chris YOUNG, responsible for porting NetSurf to Amiga systems)[9] ;
- About Odyssey, there were some attempts to port it to AmigaOS3 (including the WebKit framework)[10].

We finally decided to focus on NetSurf only (using the C language), as for Odyssey we would have to import WebKit (which did not correspond to the scope of the Ptidej Team) (see Figure 4, page 4).

---

[9] The file is available in the Appendix, as "Aymen's Guidelines". Aymen's report is available in the Bibliography.

[10] The source is available in the bibliography and named *Odyssey Web Browser: Solve PPC Webkit Javascript Engine Endianess Problems inherited from Webkit X86! - GitHub*

4.2 - Fixing the Guidelines

To create a script compiling automatically NetSurf for AmigaOS3, I could use the file created by Aymen ZALILA, a previous student who worked on this project. This file contained all the steps to compile manually NetSurf for AmigaOS3. The file was available in a Linux and a Windows version (using Cygwin). I focused on the Windows one because the computer on which I worked was only equipped with Windows.

Here are the steps to compile NetSurf for AmigaOS3:

- Download the packages needed for the following commands (for Linux only. Cygwin must be already installed with the packages) ;
- Download and compile the toolchains* ;
  - While compiling the toolchains, quickly modify manually some files to enable the NetSurf compilation with Cygwin (those files are created and quickly deleted by the toolchains compilation) ;
- Modify Linux/Cygwin PATH and PKG_CONFIG_PATH variables to add folders NetSurf and its libraries need to be compiled ;
- Compile the SDK (in the toolchains) ;
- Download & Compile the build system, then the libraries (in a precise order) and the nsgenbind (used especially to link NetSurf to its JavaScript engine named DukTape) ;
- Download & Compile NetSurf.

While following the guidelines, due to a NetSurf update including the NetSurf and DukTape combination (which caused the DukTape issue, mentioned later in this report), the guidelines did not work anymore. To make them work again, I needed to modify the Makefile[11] by adding the following line:

```
override NETSURF_USE_DUKTAPE := NO
```

This line asks the Makefile to `override`, so to erase the previous value of the `NETSURF_USE_DUKTAPE` variable, to change it to No (`:= NO`). Thus, the compilation will not use DukTape, to avoid the DukTape Issue.

[11] The file used to compile NetSurf for AmigaOS3.

There was another issue, due to the toolchains. The toolchains are quite long to compile (approximately one hour[12]). So, during the tests, Aymen advises me not to recompile them. But they had been updated between their last compilation and our tests, which led to compilation errors. It was enough to delete the folder containing all the tools (`/opt/netsurf`) to recompile the toolchains again.

Then, another problem came from Cygwin. To compile everything needed for NetSurf, some tools must be installed before. For example, we can quote `git` (the command used to download NetSurf files) and `make` (the command used to compile NetSurf files). This part is easy to do on UNIX/Linux systems, as it only needs the following command: `apt-get install git make`.

The problem is Cygwin did not have a corresponding tool. So, to install `git` and `make`, Cygwin must be reinstalled to include during its reinstallation the git and make tools (by checking boxes). Fortunately, an unofficial tool called apt-cyg was available on GitHub. This tool works as apt-get does, which simplifies the tools installation.

After the NetSurf compilation, NetSurf needed some other resource files (pictures, translations...) to work. This was not pointed out in the guidelines. These files needed to be downloaded separately. Eventually, it was enough to modify the command used to compile NetSurf to obtain the resource files needed and also a ready-to-go NetSurf package. Here is the old command followed by the new one (make is the command used to compile NetSurf):

```
make TARGET=amigaos3
make TARGET=amigaos3 package
```

After verifying the fixed guidelines were working, it was time to automatize the whole process and make it user and developer-friendly. This led to NetScript.

[12] The time taken for a compilation highly depends on the computer used to do it.

5 - Bring NetScript to life!

NetScript is the script that I created thanks to all the people quoted in the Acknowledgements part. These main goals were to automatize the NetSurf compilation process and to make it user and developer-friendly. It evolved continuously during the internship (and still evolves). It is composed of 4 main files and folders:

- NS.sh, the file containing the commands to download and compile all the files, tools, and libraries needed to compile NetSurf;
- NetScript.sh, the file used to execute NetScript, which only calls NS.sh but allow NetScript to create a log file (in case the NetSurf compilation would not work);
- updateFiles.sh, the file in charge of modifying all the files needed to compile NetSurf (for example, the NetSurf Makefile);
- The updateFile folder, containing data to modify in each file in order to enable the NetSurf compilation.

The NetScript files are available in its 20160814 version in the Appendix. You can also find the last version in my GitHub.

To automatize the files modification (especially for the files created and deleted quickly), updateFiles.sh first watch the file arrival. When the file appears, updateFiles.sh calls immediately the corresponding modification file available in the updateFile folder. Then, this file quickly modifies the part of the file needed to enable the NetSurf compilation. These steps are done directly when the file watched is created, so we avoid the manual risk to fail the compilation because the modification has been done manually and too slowly.

NetScript also asks the user some questions, to help compile NetSurf:

- Whether the user wants to compile NetSurf with DukTape (as the DukTape Issue is not fixed for the moment);
- Whether the user wants to keep the files and libraries used during the compilation, to dig in the code after. This could let the user tracking bugs in those files and libraries or modify them;
- Whether the user wants to keep the `/opt/netsurf` folder (which contains the tools used to compile NetSurf). As the previous point, this could let the user tracking bugs in it. Moreover, keeping this folder allows NetScript to speed up the whole process (as said before the toolchains compilation takes about one hour, while the files, libraries and NetSurf compilation takes approximately 20 minutes).
- Whether the user wants to clean its actual workspace (the folder where NetScript is launched). In fact, NetScript downloads the files and libraries only if they do not exist in the workspace. So, if the files exist, even if they are not updated, NetScript will not download them. This is a choice that I made because NetSurf is still in development, so somebody can easily modify the code and recompile it with the modifications.

In this version, there is also a QUICK MODE, which only recompiles NetSurf (not the libraries and files before). This mode allows NetScript to speed up again the compilation (15 minutes for the files and libraries compilation against 3-5 minutes for the NetSurf compilation only). This mode can be used only if NetScript has been launched once and if the user kept all the files and libraries in its workspace.

NetScript could be enhanced during a future project. For example, it would be interesting that NetScript compiles NetSurf for any OS (RISC OS, Windows, AmigaOS3, AmigaOS4...), on any OS (Windows, UNIX/Linux...). Also, NetScript could be more accurate to find the cause of a bad ending when a user executes it.

Now NetScript is available, it is easier to compile NetSurf for AmigaOS3 in Windows. That means we can now focus on the DukTape Issue.

6 - The DukTape Issue

As said before, DukTape is the JavaScript engine of NetSurf. It has been embedded in NetSurf 3.6dev (NetSurf was unable to interpret JavaScript content before). NetSurf 3.6dev works alone on AmigaOS3 and the same for DukTape. The issue comes when we compile NetSurf with DukTape (so, when NetSurf and DukTape are linked). Here is the issue when launching NetSurf with DukTape:

```
/handlers/javascript/duktape/dukky.c:565 js_newcontext: Creating new duktape javascript context
FATAL 56: uncaught error
PANIC 56: uncaught error (calling abort)
Abnormal program termination
```

My mission was to track the bug provoking this issue. The problem is that, unfortunately, NetSurf does not have unit tests. When debugging this issue, I really feel this lack, because I had to add everywhere manually in the suspected functions log lines, because the debug system embedded was not working yet (Chris YOUNG managed to make it work while I was ending). Functions could sometimes contain more than 500 lines of code and it was a long ordeal to log every line and to follow the way NetSurf took to obtain this issue. Here is an example of a logged function:

```
DUK_LOCAL void duk__advance_expect(duk_compiler_ctx *comp_ctx, duk_small_int_t expect){
	LOG("duk__advance_expect() BEGIN");
	LOG("duk__advance_expect() : duk__advance_helper(comp_ctx, expect);");
	duk__advance_helper(comp_ctx, expect);
	LOG("duk__advance_expect() END");
}
```

Also, there are sometimes choices took by the developers for their code that lessen its readability and understandability. For example, the function above only calls another function. Moreover, it has a name with two underscores (__) as almost every other function (we could think first this function was special). We can also find some infinite “for” loops (with no conditions), where the code gets out by using breaks and long jumps, which is not a good practice. Maybe the code needs a refactoring.

To debug the code, the routine was the same:

- Look the LOG file obtained from the NetSurf execution and locate the function before the FATAL 56 line;
- Locate this function into the DukTape files;
- Log this function completely, one row after another;
- Compile NetSurf;
- Execute NetSurf in AmiKit and return to the first step.

After a while, the LOG file revealed an interesting thing.... Here it is (this part contains ellipses and is simplified. The entire raw part is available in the Appendix):

```
duk_js_compiler.c:duk__advance_helper() BEGIN
[…]
	if (expect >= 0 && comp_ctx->curr_token.t != expect) BEGIN
		DUK_D(DUK_DPRINT(parse error: expect=315438081, got=290994208, (long) expect,
		(long) comp_ctx->curr_token.t));
		DUK_ERROR_SYNTAX(thr, DUK_STR_PARSE_ERROR);

duk_error_longjmp.c:duk_err_longjmp()BEGIN
[…]
	if (!thr->heap->lj.jmpbuf_ptr) BEGIN
	if (!thr->heap->lj.jmpbuf_ptr) END
[…]
[We indirectly return to duk_js_compile(), the function which previously indirectly calls
duk__advance_helper()]

[We are in duk_js_compiler.c:duk_js_compile()]
	safe_rc = 1;
	thr->compile_ctx = prev_ctx;
	if (safe_rc != DUK_EXEC_SUCCESS) BEGIN
		duk_throw(ctx);

duk_api_stack.c:duk_throw() BEGIN
[…]
	duk_api_stack.c:4356 duk_throw: duk_throw() : duk_err_longjmp(thr);

duk_error_longjmp.c:9 duk_err_longjmp: duk_err_longjmp() BEGIN
[…]
	if (!thr->heap->lj.jmpbuf_ptr) BEGIN
		DUK_D(DUK_DPRINT(uncaught error: type=290994208 iserror=315475192 value1=!T
		value2=!T, (int) thr->heap->lj.type, (int) thr->heap->lj.iserror, &thr->heap-
		>lj.value1, &thr->heap->lj.value2));
		duk_fatal((duk_context *) thr, DUK_ERR_UNCAUGHT_ERROR, uncaught error);

duk_api_stack.c:duk_fatal() BEGIN
[…]
	thr->heap->fatal_func(ctx, err_code, err_msg);

FATAL 56: uncaught error
PANIC 56: uncaught error (calling abort)
Abnormal program termination
NetSurf : erreur code 20
```

As we can see here, the error seems to appear because of a long jump due to a syntax error. After Yann-Gaël and I saw this, we climbed back up to this piece of code, located in the function duk__parse_func_like_fnum(), in duktape.c:

```
duk__advance_expect(comp_ctx, DUK_TOK_RCURLY);
```

So, we supposed at this point that the issue was due to a JavaScript test page (the NetSurf welcome page does not contain JavaScript), executed during the NetSurf launch, to test whether DukTape works. Unfortunately, we could not verify this piece of information as I did not find the suspected DukTape JavaScript test page in the code and as we did not receive any answer about that from the developers yet (except Chris Young, who told me to ask the other developers because he did not know).

To put things into perspective, if this conjecture is proved to be the real source of the DukTape issue, that means DukTape crashes and make NetSurf crash too only because of a little mistake in a test Web page. We can wonder why the developers do not decide to deactivate DukTape, log an error and continue loading the page in case the DukTape tests do not work, to let NetSurf work without it.

7 - Spread NetSurf to the RasPi World!

7.1 - Raspberry Pi?

Figure 17 - A Raspberry Pi 2

The Raspberry Pi is a credit-card-sized micro-computer created in 2012 by the Raspberry Pi Foundation. It was first intended to help kids to learn coding and Computer Science at School (thanks to its low cost and its UNIX/Linux distribution called Raspbian) It quickly became a very famous smart object (belonging to the Internet of Things) all over the Computer Science world, used by several beginners to develop their coding skills and design some projects: gaming console, media center, smart mirror, automaton, robot... There are now different versions of Raspberry PI (for example, the most known Raspberry Pi 2, or the Raspberry Pi 0, the smallest one to embed it easily.

7.2 - Happiga?

Figure 18 - The Happiga Logo

During the project, I used my own Raspberry Pi 2 with the Happiga distribution. Happiga is a UNIX/Linux based OS which uses the sources of the Amiga emulator (called UAE4ARM) to emulate an Amiga computer in the Raspberry Pi. Happiga was created by BPJ Studio (Erwan JESTIN) in 2016 and was first a range of Happi, the first BPJ Studio OS, used to play retro games (including Amiga games). As an Amigan, Erwan decided to design Happiga, because he wanted young and actual generations to (re)discover it easily by making it available on the low-cost Raspberry Pi. Happiga is not forcefully user-friendly, but a tutorial exists to get started with it. I suggested Yann-Gaël to import NetSurf to Happiga, because it corresponded to what Yann-Gaël was interested in and also because a Web Browser is quite important in an OS with the power of Internet and IoT nowadays. Porting NetSurf to Happiga will maybe encourage curious people to discover Happiga and so the Amiga world.

7.3 - NetSurf on Happiga

As Happiga can emulate an AmigaOS3 OS, it can also launch NetSurf. That means there is no difference between compiling NetSurf for Happiga and for AmiKit. However, these two systems are not on the same computer, and so do not enjoy the same hardware. As the Raspberry Pi as "only" 1 GB RAM memory (which is basically good enough for Happiga) and NetSurf has a memory leak (it is still in development), it is more difficult to make it work on Happiga.

The first problem was to find a way to move the NetSurf archive (obtained from NetScript) into a folder reachable by Happiga. Fortunately, Happiga has some shared folders, accessible via the network.

The next problem was to install NetSurf into Happiga. Indeed, we must get an installer compatible with the Workbench available on Happiga. The last Workbench version available for Amiga was the Workbench 3.1 one, so the last available OS version is AmigaOS 3.1. The next versions (AmigaOS 3.5 and 3.9) are not officially available because of the Amiga bankruptcy (which came before their open release).

The installer was easy to solve thanks to the command we talked before:

```
make TARGET=amigaos3 package
```

By creating a package, the compilation automatically creates an install file for AmigaOS 3.5 and above. So, the problem was to find a way to obtain minimum AmigaOS 3.5. I finally found AmigaOS 3.9 thanks to Erwan JESTIN (Happiga Creator), and could install NetSurf on Happiga.

After installing NetSurf (without DukTape, as the DukTape issue is not solved yet), NetSurf did not want to start for two reasons. At first, two libraries (used for the Internet TCP connection) were missing: bsdsocket and usergroup. The solution here was to add those libraries to the "Libs" AmigaOS folder. As Erwan already implemented the bsdsocket one in a new version of Happiga, I gave the usergroup one for a future update.

About the second point, the cause was again the NetSurf memory leak. The solution was to modify the "Choices" files, created during the NetSurf installation and located in the User/Default folder. I needed to add this line to the file:

```
disc_cache_size:0
```

This line means we will not use cache memory to use NetSurf, due to its memory leak and the fact Happiga has only 1GB RAM memory for leeway.

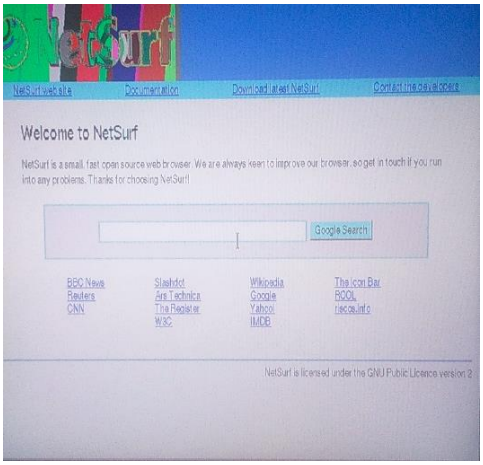


Figure 19 - The NetSurf welcome page during its first start on Happiga

After some attempts, NetSurf finally started.

Now NetSurf worked on Happiga, there were two others issues to solve, inherent to Happiga (so, not the memory leak): displaying pictures and accessing SSL Encrypted Web Pages (Amazon, Wikipedia...).

About the display issue, I first thought the rainbow effect visible on the pictures (see Figure 19) was due to the lack of datatypes*, but finally the problem was due to NetSurf, which did not support the Happiga display settings to display. This has been settled by a NetSurf update.

The SSL encrypted Web pages issue is a real problem, as the most used part of Internet will soon become totally protected by SSL encryption (it is already the case for Wikipedia and Google). NetSurf owned the needed certificates to access those websites, but it seemed it did not use them. Compared to AmiKit (the Amiga emulator on Windows), Happiga did not have the AmiSSL tools installed. By installing them, I managed to access Wikipedia once, but not the other Web pages. It would be interesting to focus on that for a future project.

Figure 20 - NetSurf display test BEFORE (TOP) and AFTER (BOTTOM) the update

8 - Project Management

I did my internship from June 6th to August 28th in the Computer Science and Software Engineering Department at École Polytechnique de Montréal, in Canada. During this period, it was easy to go to work (there was no snow), but a problem is that most of the people were in holidays, so the Department worked slowly.

As you will read further on, I used many easy-to-use tools to work on this project:

- My Raspberry Pi a micro-computer belonging to the Internet of Things. It's a pity that they did not have one in the lab, fortunately I had mine ;
- Happiga, the Amiga Distribution for Raspberry Pi ;
- Notepad++ (a text editor) ;
- NetScript (of course), the script I created to compile easily and quickly NetSurf for AmigaOS3 ;
- Shell (to write and test NetScript) ;
- AmiKit, an Amiga emulator on Windows, which helped me to test and debug NetSurf. This tool often crashed during its execution, for no reason I could figure out ;
- My own computer and those of PolyMTL (containing Aymen's guidelines).

To be in contact with Yann-Gaël, my PolyMTL tutor, I often used e-mails (one per day) and face to face when he was available. He answered quickly (once a day, or in a couple of days), except during its holidays (from June 19th to July 20th), when it was more difficult to join him because he did not have any network there. I often exchanged by e-mail with Vincent BERRY, my Polytech tutor too (once a week), and we managed to do some video calls despite of the time difference. I also often talked with Erwan JESTIN (Happiga creator) by Facebook Messenger and Chris YOUNG (responsible for porting NetSurf to Amiga systems) by mail and by the amiga.org forums.

It was easy to manage my work as I was not in a team (my project will lead to a future team project for the Ptidej team). In the Gantt diagram below, some tasks are not completely achieved. Here are the explanations:

- NetScript : NetScript could be enhanced and updated in a future project, so we cannot say these steps are achieved (a piece of software is never totally finished unless the development is stopped) ;
- DukTape Issue - Locate the issue: we are not sure the issue is located where we supposed, so we cannot really say the Issue is due to the quoted piece of code.
- DukTape Issue – Warning NetSurf Developers: I did not receive any answer from them, so I do not know if they are warned about ;
- NetSurf Happiga – Debugging: As you will read, the "SSL Issue" is not fixed yet.

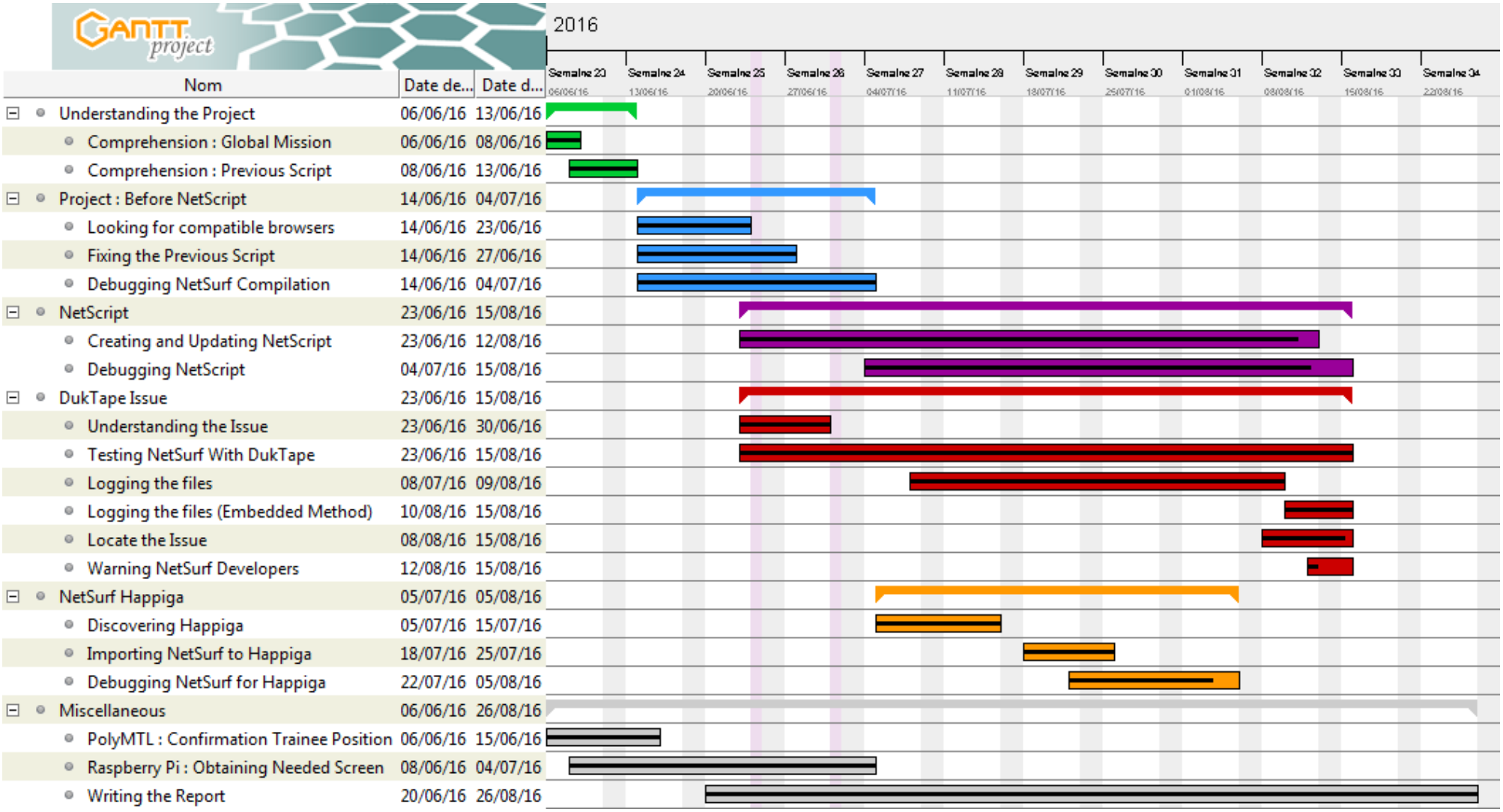


Figure 21 - My Gantt diagram (from June 6th to August 28th)

9 - Conclusion

The first part of my mission was to find a browser which could be compatible with AmigaOS3 and to compile it for. The second part was focused on NetSurf: fixing the guidelines used to compile it for AmigaOS3, debugging it and then porting it to Happiga.

Eventually, I did not find any other Web browser than NetSurf to be compiled for AmigaOS3. A good pretender was Odyssey, but it uses WebKit, which has not been ported to AmigaOS3 yet. So, we leave it because this work did not correspond to the future Ptidej project Yann-Gaël wanted to create. Maybe WebKit will be imported in a short time, so the future student working on this project will probably have to take a look at Odyssey Web Browser again.

About Aymen's guidelines, I managed to fix them. It leaded me to create NetScript, a script to compile NetSurf for AmigaOS3 automatically, easily and quickly on Windows (via Cygwin). This script is efficient now and was used by some amiga.org forum users (3 to 5 people for the moment) to help the developers to debug NetSurf (Display issues, the DukTape Issue, Memory leaks...), and we approximately exchanged 50 messages about[13]. However, NetScript could be enhanced by another student during a future project. For example, it could compile NetSurf for any OSes (RISC OS, UNIX/Linux, Windows, AmigaOS4...), on any OSes (Windows, UNIX/Linux...), or someone could enhance the way to catch compilation errors.

About the NetSurf debugging, I focused on the DukTape Issue. I think Yann-Gaël and I managed to get closer to the issue (a JavaScript test page with some code mistakes), but we cannot be sure about that. It could be a great idea to re-contact NetSurf developers to have an idea about this conjecture.

Finally, about porting NetSurf to Happiga, I managed to do it thanks to Happiga designer. Now we know how to make NetSurf work on Happiga. However, NetSurf is not totally stable due to its memory leak (not only on Happiga). It could be interesting that a future student focuses on the SSL Encrypted Web pages Issue (specific to Happiga), as Web pages will use more and more this type of encryption.

I was really happy to discover the often forgotten Amiga World. I could understand the problems about linking an old-school OS with a modern Web browser (especially the compatibility issues) I could enhance my Shell and Management skills thanks to NetScript and this project. I could also realize we cannot remember and know totally all the rules of a coding language (here, the C one). We must maintain our coding skills to not lose them.

[13] The source is available in the Bibliography and is called *duktape error NetSurf OS3 - Amiga.org*.

10 - Glossary

*In the report, the words with an asterisk * are explained here (only the first word occurrences have an asterisk).*

Bit

The bit is the smallest piece of information possible in a computer and can be equal to 0 or 1. Several bits can be united to become a data. For example 0010 1010 is an 8-bit binary number equals to the decimal number 42 and which could correspond to the RED color data 6, 8

Cygwin

Cygwin is a Shell emulator (based on UNIX/Linux systems) for Windows 3, 14, 15, 26, 40, 42, 52, 54, 56

Datatype

a datatype is a file used to explain how AmigaOS has to display a specific type of picture, as PNG, GIF, JPG... 23

Endianness

Endianness is the name given to a couple of rules followed by an architecture and describing the way to read data (composed of bits). For example, with the data 101010, an architecture following the Big Endian rule will read the data from left to right (starting from the 1) while an architecture following the Small Endian rule will read the data from wright to left (starting from the 0) 13

Framework

A framework is a set of tools used to make the development of a piece of software easier and to bring it useful features 5

Processor

A processor could be compared to the computer's brain, except for memory. It is responsible for transmitting and calculating the data needed to make a software and globally the computer work 6

Retro gamers

A retro gamer is gamer who likes playing old videogames (~10+ years old)) 11

SSL

SSL, for Secure Sockets Layer, is a Web pages encryption technology to avoid hacking 11, 26, 30

Text-based OS

A text-based OS is an operating system which does not contain any graphical element on its User Interface. To do an action, you have to write one or several command line(s). This type of OS can be powerful, at the cost of the user-friendliness. For example, MS-DOS is a text-based OS, belonging to the DOS text-based OS family 7

Third Party company

A third party company, is a company which has no link with the device we talk about (for example here the first Amiga device) and which is not owned by the company developing this device (in opposition to a first party company. However, this company develops some things for it. For example, Super Mario Bros is a video game series developed by a first party company for the Nintendo game consoles (it is Nintendo itself) whereas Rayman is a video game series developed by a third party company for the Nintendo game consoles (it is Ubisoft, not Nintendo, which produces these games) 6

Toolchains

The toolchains are the tools used to link every library to NetSurf and to compile NetSurf properly 14, 15, 17, 37, 42, 43, 51, 52, 53, 56, 57, 58

11 - Bibliography

11.1 - Documents

[Picture] Amiga New Logo. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/File:Recent_Amiga.svg

[Picture] Amiga Old Logo. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/File:Amiga_Logo_1985.svg

[Picture] Apple Safari Logo. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Fichier:SafariOSXYosemite.png

[Picture] AWeb Logo. (n.d.). Retrieved from Greyhound Data: http://www.greyhound-data.com/gunnar/aweb/

[Picture] Binary Code. (n.d.). Retrieved from Flaticon: http://www.flaticon.com/free-icon/binary-code_76318

[Picture] Ditto (Metamorph). (n.d.). Retrieved from Pokémon Trash: http://www.pokemontrash.com/pokedex/images/sugimori/132.png

[Picture] Google Chrome Logo. (n.d.). Retrieved from Wikimedia Commons: https://commons.wikimedia.org/wiki/File:Google_Chrome_for_Android_Icon_2016.svg

[Picture] Happiga Logo. (n.d.). Retrieved from BPJ Studio - Facebook: https://www.facebook.com/bpjstudio/

[Picture] IBrowse Logo. (n.d.). Retrieved from IBrowse: http://www.ibrowse-dev.net/

[Picture] Internet. (n.d.). Retrieved from Flaticon: http://www.flaticon.com/free-icon/internet_149229

[Picture] Juggler Animation. (n.d.). Retrieved from Obsolete Technology Website: http://www.oldcomputers.net/amiga1000.html

[Picture] Musica Searcher. (n.d.). Retrieved from Flaticon: http://www.flaticon.com/free-icon/musica-searcher_70376

[Picture] NetSurf Logo. (n.d.). Retrieved from Wikimedia Commons: https://commons.wikimedia.org/wiki/File:NetSurf-logo.svg

[Picture] NetSurf Logo (complete). (n.d.). Retrieved from NetSurf git repositories: http://source.netsurf-browser.org/art.git/plain/logo/full/NetSurf.svg

[Picture] Odyssey Logo. (n.d.). Retrieved from Power2People: https://power2people.org/images/odyssey_logo_large.png

[Picture] Playstation. (n.d.). Retrieved from Affaire de Gars: http://www.affairesdegars.com/webroot/usr_img/2014-12/55912830/playstation_3.jpg

[Picture] Raspberry Pi 2. (n.d.). Retrieved from StaticWorld.net: http://core0.staticworld.net/images/article/2015/02/raspberry-pi-2-sd-card-100569129-orig.png

[Picture] Timberwolf Logo. (n.d.). Retrieved from AmigaOS.net: http://www.amigaos.net/sites/default/files/slide-2_0_0_0.png

[Picture] Workbench 1.0. (n.d.). Retrieved from Greg Donner: http://www.gregdonner.org/workbench/wb_10.html

[Report] Portage d'un navigateur web sur AmigaOS 3.x, Aymen ZALILA. (Winter 2016). Retrieved from http://bit.ly/2bdoDX4

[Report] Proposed Research Program, Yann-Gaël GUEHENEUC. (n.d.). Retrieved from http://bit.ly/2b1HNig

[Text File] AWeb Public License. (n.d.). Retrieved from Yvon Rozijn's Web Page: http://www.yvonrozijn.nl/aweb/apl.txt

[Video] Amiga Juggler 3D Animation - YouTube. (n.d.). Retrieved from YouTube: https://www.youtube.com/watch?v=-yJNGwIcLtw

11.2 - Web Pages

A-EON Technnology. (n.d.). Retrieved from A-EON Technnology: http://www.a-eon.com/

Amiga - Wikipedia. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Amiga

Amiga History. (n.d.). Retrieved from Amiga History Guide: http://www.amigahistory.co.uk/

AmigaOS - Wikipedia. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/AmigaOS

AmigaOS.net. (n.d.). Retrieved from AmigaOS.net: http://www.amigaos.net/

Atari History. (n.d.). Retrieved from Atari: https://www.atari.com/history/1972-1984-0

AWeb. (n.d.). Retrieved from Greyhound Data: http://www.greyhound-data.com/gunnar/aweb/

Commodore 64 - The Best Selling Computer In History. (n.d.). Retrieved from Commodore: http://www.commodore.ca/commodore-products/commodore-64-the-best-selling-computer-in-history/

Commodore Amiga 1000. (n.d.). Retrieved from Obsolete Technology Website: http://www.oldcomputers.net/amiga1000.html

Commodore Amiga CD 32 (1993 - 1995). (n.d.). Retrieved from Gros Pixels: http://www.grospixels.com/site/cd32-1.php

Cygwin - Wikipedia. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Cygwin

deadwood-pl/OdysseyWebBrowser - GitHub. (n.d.). Retrieved from GitHub: https://github.com/deadwood-pl/OdysseyWebBrowser

DNADNL/NetScript - GitHub. (n.d.). Retrieved from GitHub: https://github.com/DNADNL/NetScript

duktape error NetSurf OS3 - Amiga.org. (n.d.). Retrieved from Amiga.org: http://www.amiga.org/forums/showthread.php?t=70612

EyMenZ/NetSurf-OS3 - GitHub. (n.d.). Retrieved from GitHub: https://github.com/EyMenZ/NetSurf-OS3

Framboise 314. (n.d.). Retrieved from Framboise 314: http://www.framboise314.fr/

Framework - Wikipedia. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Software_framework

Happiga - Happi Game Center. (s.d.). Récupéré sur Happi Game Center: http://happi-game-center.com/

IBrowse. (n.d.). Retrieved from IBrowse: http://www.ibrowse-dev.net/

Linux - Wikipedia. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Linux

Motorola 68000 - Architecture et Programmation. (n.d.). Retrieved from Eagle System: http://eaglesystem.free.fr/68000.html

Motorola 68000 - Wikipedia. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Motorola_68000

NetSurf git repositories. (n.d.). Retrieved from NetSurf git repositories: http://source.netsurf-browser.org/

Odyssey Web Browser. (n.d.). Retrieved from Power 2 People: https://www.power2people.org/projects/odyssey/

Odyssey Web Browser: Solve PPC Webkit Javascript Engine Endianess Problems inherited from Webkit X86! - GitHub. (n.d.). Retrieved from GitHub: https://github.com/deadwood-pl/OdysseyWebBrowser/issues/1

Playstation - Wikipedia. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/PlayStation_(console)

Raspberry Pi. (n.d.). Retrieved from Raspberry Pi.org: https://www.raspberrypi.org/

Raspberry Pi - Wikipedia. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Raspberry_Pi

Web of the Ptidej Team. (n.d.). Retrieved from Web of the Ptidej Team: http://www.ptidej.net/

Where to get a full version of AWeb? - Amiga.org. (n.d.). Retrieved from Amiga.org: http://www.amiga.org/forums/showthread.php?t=64357

YouTube est désormais presque entièrement chiffré en HTTPS - BeGeek. (n.d.). Retrieved from BeGeek: http://www.begeek.fr/youtube-desormais-presque-entierement-chiffre-https-211505

12 - Appendix

12.1 - AmigaOS3 Web Browsers Comparison

12.1.1 - Features Comparison

This table compares the basic features we can find in an actual Web Browser. Google Chrome, which is not available in AmigaOS3, is our reference. These tests have been done in AmiKit v8 (the Amiga Emulator), using the Workbench 3.9.

Here are the features:

- Does the Web Browser have a Bookmarks Manager? (Yes/No)
- Does the Web Browser have Shortcuts (like Refresh, Home...)? (Yes/No)
- Can we look for words in a specific Web page? (Yes/No)
- Does the Web Browser have an embedded Search Engine like Google or Bing (in the navigation bar) to search Web Pages? (Yes/No)
- Is the Web Browser free to use? (Yes/No)
- Can the Web Browser access to SSL Encrypted Web pages? (Yes/No)
- Does the Web Browser have a Tab Manager? (Yes/No)
- Is the Web Browser still updated? (Yes/No)
- Can the Web Browser launch videos (on YouTube)? (Yes/No)
- Can the Web Browser read JavaScript Web pages? (Yes/No)

Here are the results (see the notes for more information):

	Chrome 52	NetSurf 3.6dev	AWeb 3.5	IBrowse 2.4
Bookmarks	Yes	Yes	Yes	Yes
Shortcuts	Yes	Yes	Yes	Yes
Looking for Words	Yes	Yes (1)	Yes	Yes
Search Engine	Yes	No	Yes	Yes (2)
Free	Yes	Yes	Yes	No
SSL Web pages	Yes	Yes	No (3)	No
Tab Manager	Yes	Yes	No	No
Still Updated	Yes	Yes	No	No
Videos	Yes	No	No	No (4)
JavaScript	Yes	No	No	No

Notes :

(1) The system is embedded but not totally functional (Typing Issue).
(2) The Search Engine isn't located in the navigation bar but in a specific sidebar.
(3) AWeb doesn't recognize the SSL encryption, but lets the user choose to go (or not) to these Web Pages.
(4) IBrowse cannot open YouTube because it's an SSL Encrypted Web Page.

12.1.2 - Display Comparisons

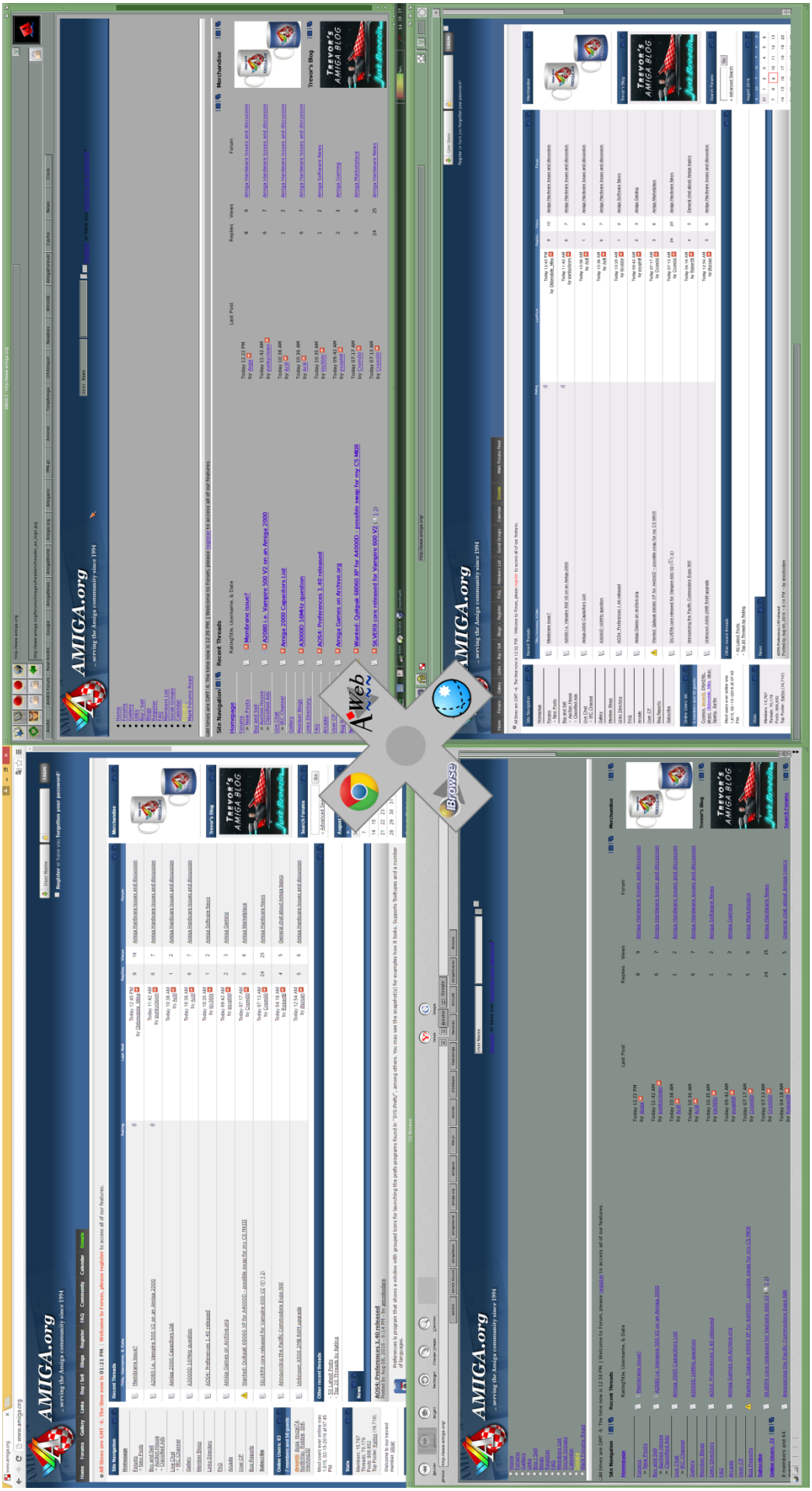


Figure 22 - Display Comparisons: Amiga.org

Figure 23 - Display Comparisons: Google

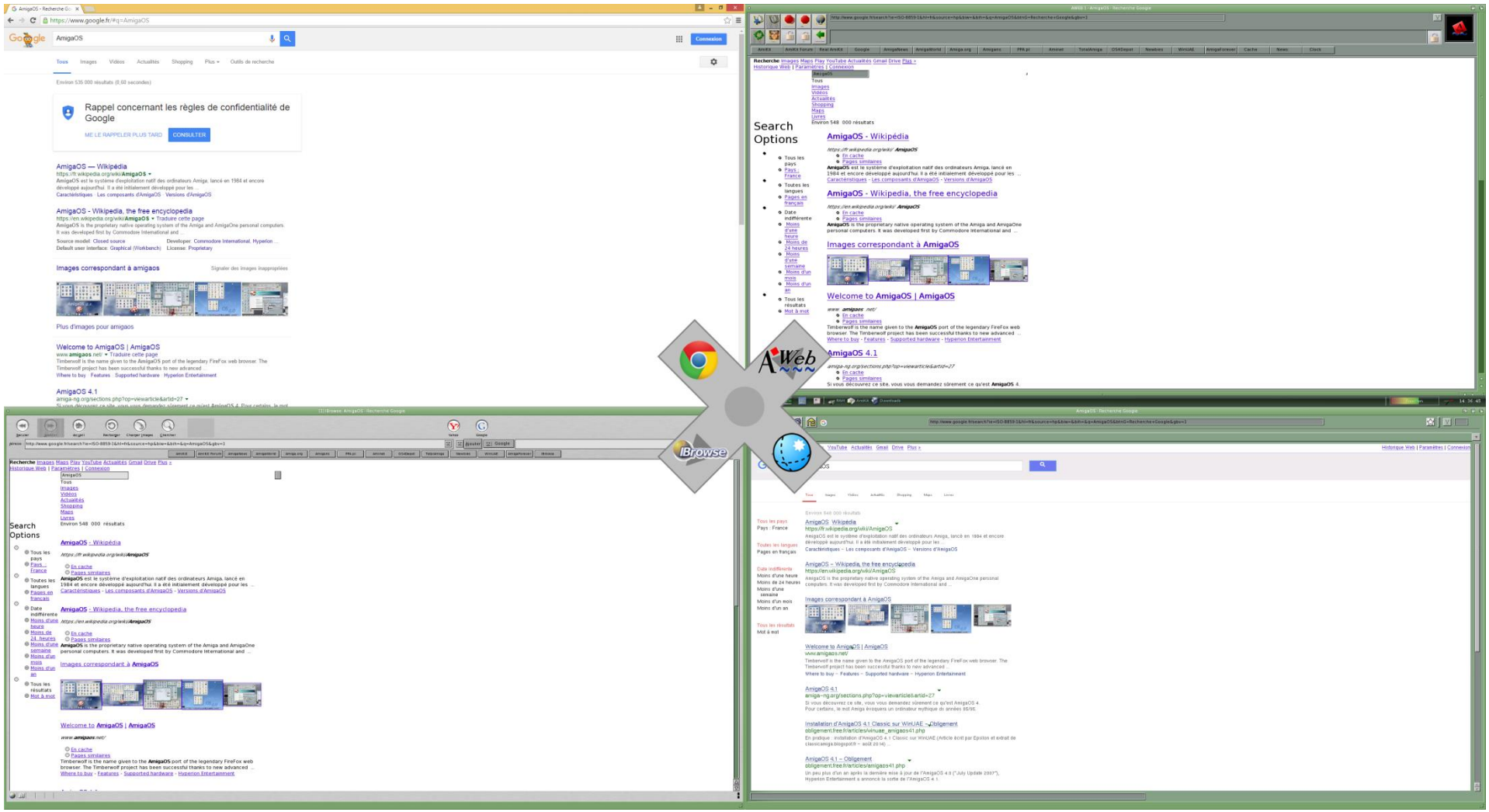


Figure 24 - Display Comparisons: Searching AmigaOS on Google

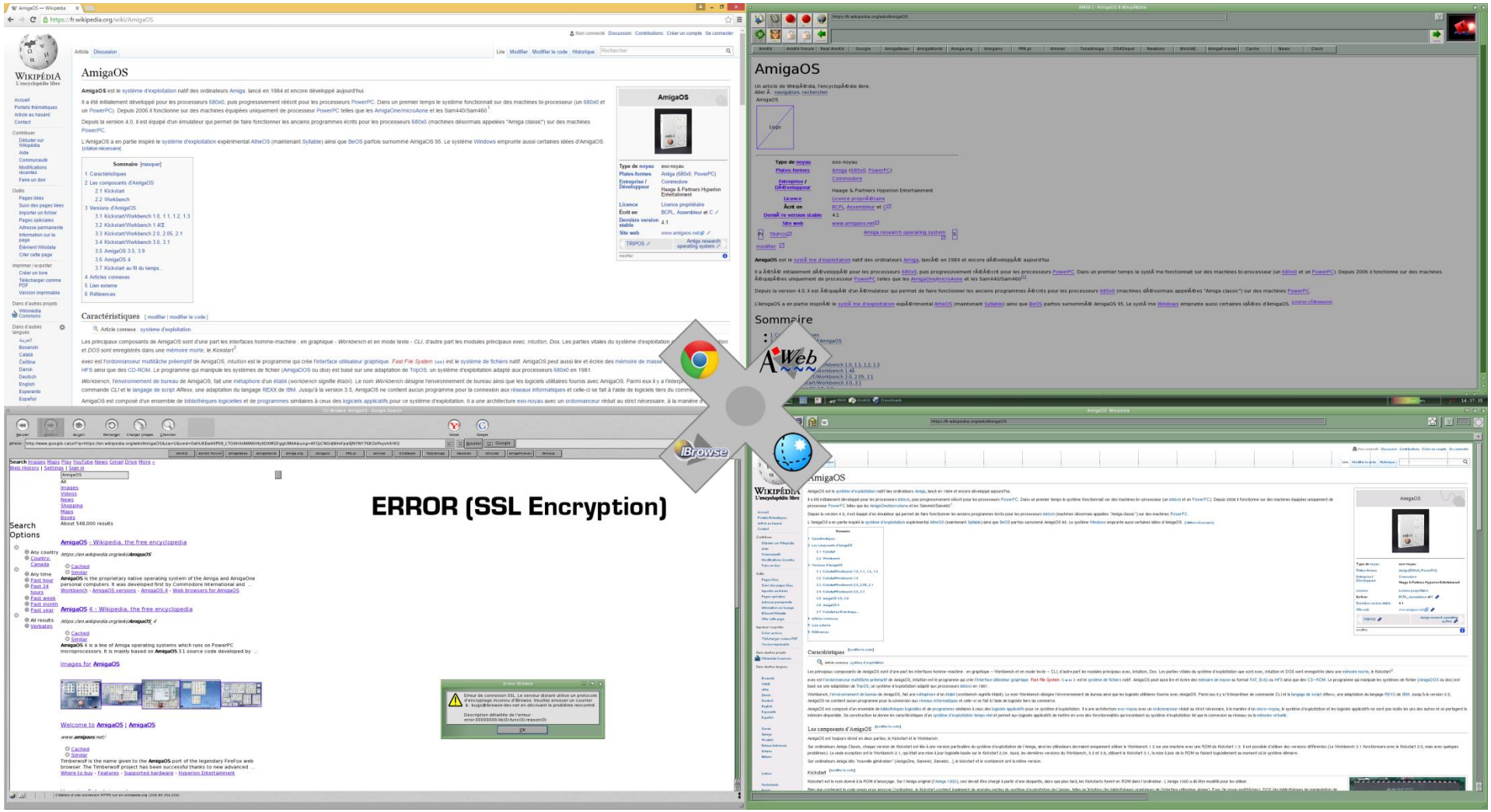


Figure 25 - Display Comparisons: AmigaOS on Wikipedia

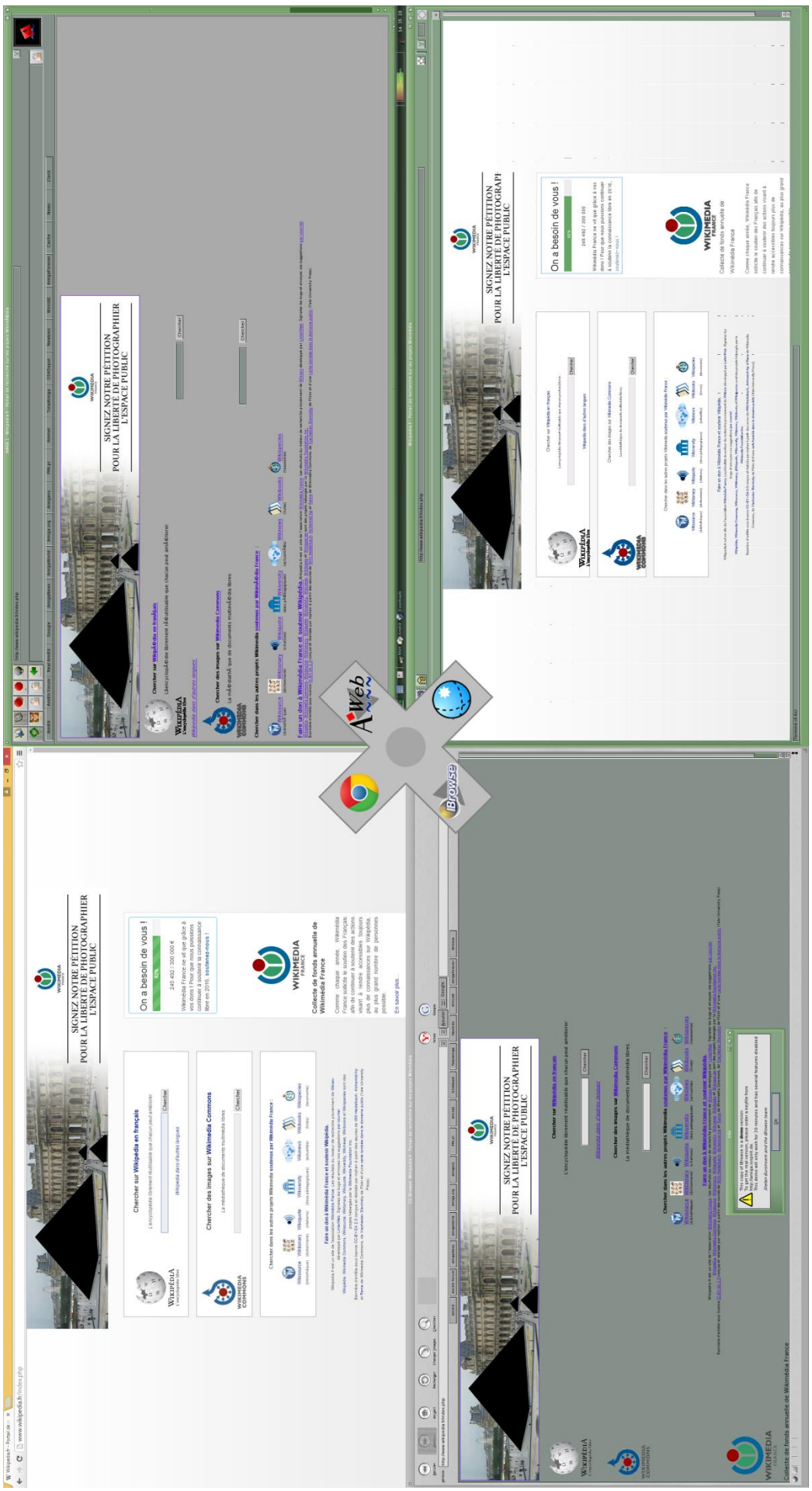


Figure 26 - Display Comparisons: Wikipedia Home Page

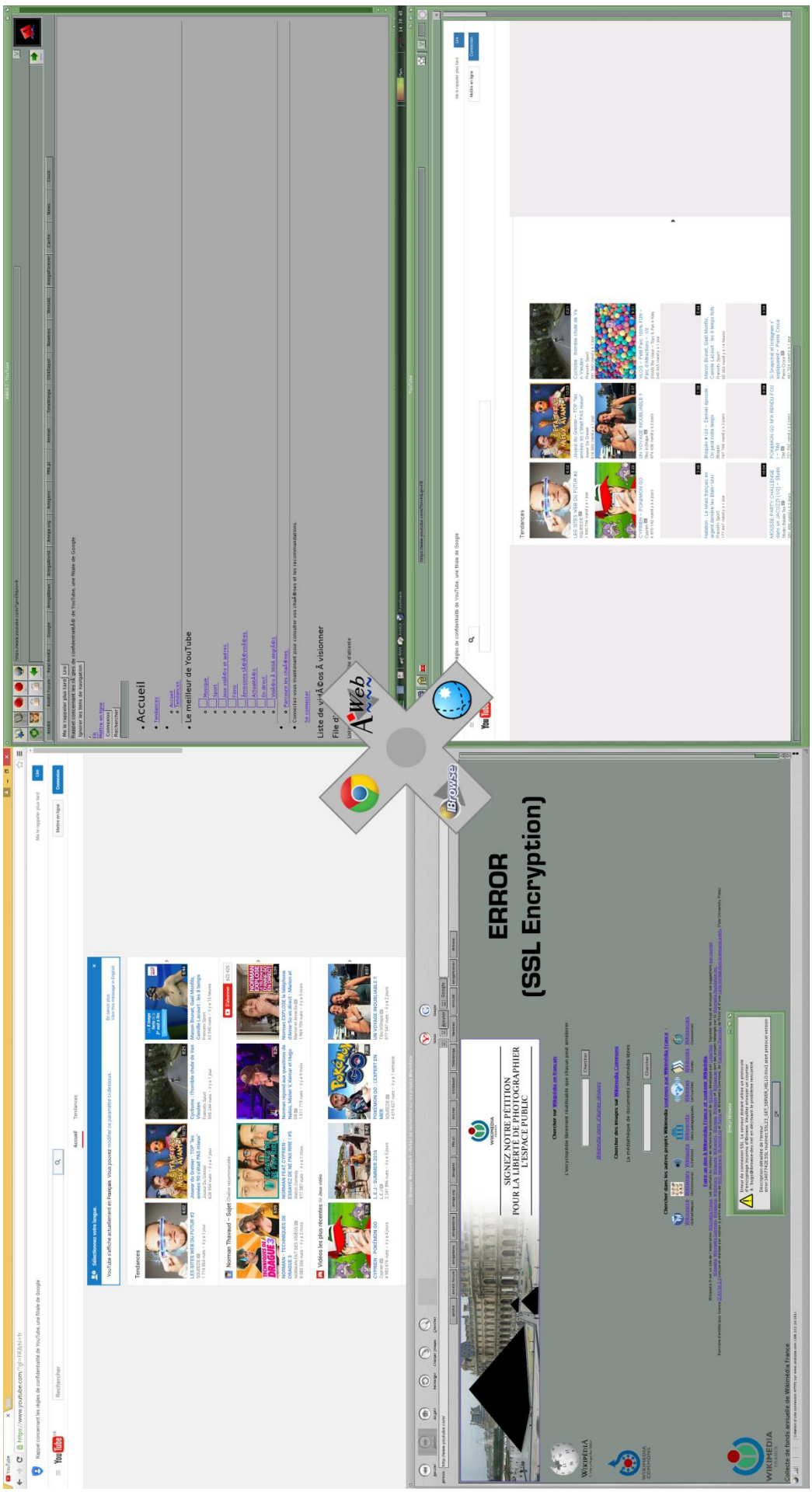


Figure 27 - Display Comparisons: YouTube

12.2 - Aymen's Guidelines

```
sudo apt-get install apache2 g++ python subversion gperf make devscripts fakeroot flex bison
autoconf autoconf2.64
ln -s /opt/gcc-tools/epoch2/bin/autom4te-2.64 /usr/bin/autom4te2.64
ls -al /usr/bin/autom4te*

git clone git://git.netsurf-browser.org/toolchains.git
cd toolchains/m68k-unknown-amigaos
make (pendant le make: c-parse.in YYLEX -> yylex() et collect2.c ligne 1594 redir_handle = open
(redir, O_WRONLY | O_TRUNC | O_CREAT, 0666);)

creer le /opt/netsurf/m68k/unknown/amigaos/cross/bin/m68k/unknown/amigaos

export PATH=/usr/local/amiga/bin:$PATH
export PATH=/opt/netsurf/m68k-unknown-amigaos/cross/bin:$PATH
export PATH=/opt/netsurf/m68k/unknown/amigaos/cross/bin/m68k/unknown/amigaos:$PATH
export   CPPFLAGS="-I/home/Administrator/Links/zlib-1.2.8   -I/home/Administrator/Links/libpng-
1.2.56 -I/home/Administrator/Links/jpeg-9b -I/home/Administrator/Links/tiff-v4.0.6/libtiff"
export    LDFLAGS="-L/home/Administrator/Links/zlib-1.2.8    -L/home/Administrator/Links/libpng-
1.2.56 -L/home/Administrator/Links/jpeg-9b -L/home/Administrator/Links/tiff-v4.0.6/libtiff"
export PKG_CONFIG_PATH=/opt/netsurf/m68k-unknown-amigaos/env/lib/pkgconfig

modifier make les cc_ #Tools

cd toolchains/sdk
GCCSDK_INSTALL_CROSSBIN=/opt/netsurf/m68k-unknown-amigaos/cross/bin
GCCSDK_INSTALL_ENV=/opt/netsurf/m68k-unknown-amigaos/env make

CC=m68k-unknown-amigaos-gcc LD=

git clone git://git.netsurf-browser.org/buildsystem.git
cd buildsystem
git pull
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos clean
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos install
cd ..

git clone git://git.netsurf-browser.org/libnsgif.git
cd libnsgif
git pull
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos clean
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos install
cd ..
```

```
git clone git://git.netsurf-browser.org/libnsbmp.git
cd libnsbmp
git pull
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos clean
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos install
cd ..

git clone git://git.netsurf-browser.org/libwapcaplet.git
cd libwapcaplet
git pull
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos clean
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos install
cd ..

git clone git://git.netsurf-browser.org/libparserutils.git
cd libparserutils
git pull
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos clean
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos install
cd ..

git clone git://git.netsurf-browser.org/libhubbub.git
cd libhubbub
git pull
rm -Rf examples
rm -Rf perf
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos clean
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos install
cd ..

git clone git://git.netsurf-browser.org/libdom.git
cd libdom
git pull
rm -Rf examples
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos clean
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos install
cd ..
```

```
git clone git://git.netsurf-browser.org/libcss.git
cd libcss
git pull
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos clean
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos install
cd ..

git clone git://git.netsurf-browser.org/libsvgtiny.git
cd libsvgtiny
git pull
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos clean
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos install
cd ..

git clone git://git.netsurf-browser.org/libnsutils.git
cd libnsutils
git pull
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos clean
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos install
cd ..

git clone git://git.netsurf-browser.org/libutf8proc.git
cd libutf8proc
git pull
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos clean
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos
make  TARGET=amigaos3  BUILD=release  PREFIX=/opt/netsurf/m68k-unknown-amigaos/env  HOST=m68k-
unknown-amigaos install
cd ..

git clone git://git.netsurf-browser.org/netsurf.git
cd netsurf/
git pull
export PKG_CONFIG_PATH=/opt/netsurf/m68k-unknown-amigaos/env/lib/pkgconfig
make TARGET=amigaos3 clean
******** Modifier line 254 et 511 du Makefile
make TARGET=amigaos3
```

12.3 - NetScript Version 2016-08-14

12.3.1 - NS.sh

```
NETSURF_VERSION="3.6dev"
NETSCRIPT_DATE="20160814-1"

if [ ! \( "$1" = "-q" -o "$1" = "--quick" \) ]
then
	echo "                                                "
	echo "------------------------------------------------"
	echo "         NetScript for Netsurf $NETSURF_VERSION"
	echo "Crossed Compilation Cygwin (Windows) - AmigaOS3"
	echo "                                                "
	echo "       Created by DNADNL, EyMenZ & Tygre        "
	echo "                                                "
	echo "               A big thanks to :                "
	echo "          Chris Young, transcode-open           "
	echo "           and the amiga.org forums !           "
	echo "                                                "
	echo "     *** Script Version : $NETSCRIPT_DATE ***   "
	echo "------------------------------------------------"
	echo "                                                "

	while [ \( "$compileWithDukTape" != "Y" -a "$compileWithDukTape" != "N" \) -a \( "$compileWithDukTape" != "y" -a "$compileWithDukTape" != "n" \) ]
		do read -p "(1/4) Do you want to compile NetSurf WITH DukTape ?(Y/n) : " compileWithDukTape
	done

	while [ \( "$keepFiles" != "Y" -a "$keepFiles" != "N" \) -a \( "$keepFiles" != "y" -a "$keepFiles" != "n" \) ]
		do read -p "(2/4) Do you need to KEEP the files and libraries used for the compilation AFTER the NetScript execution (to dig in the code) ? (Y/n) : " keepFiles
	done

	while [ \( "$keepOptNetSurf" != "Y" -a "$keepOptNetSurf" != "N" \) -a \( "$keepOptNetSurf" != "y" -a "$keepOptNetSurf" != "n" \) ]
		do read -p "(3/4) Do you want to KEEP the /opt/netsurf folder at the end of NetScript execution ? If you keep it, the future NetScript executions will be shorter. (Y/n) : " keepOptNetSurf
	done

	while [ \( "$cleanWorkspace" != "Y" -a "$cleanWorkspace" != "N" \) -a \( "$cleanWorkspace" != "y" -a "$cleanWorkspace" != "n" \) ]
		do read -p "(4/4) Do you want to CLEAN your Workspace folder $(PWD) (that means deleting the old files and libraries) BEFORE the NetScript execution ? If you clean it, NetScript will delete the actual files and libraries and download the fresh ones, so you will lose your modifications (if you did some). (Y/n) : " cleanWorkspace
	done
```

```
if [ "$compileWithDukTape" = "Y" -o "$compileWithDukTape" = "y" ]
       then
              echo "                                                "
              echo "------------------------------------------------"
              echo "   You chose to compile NetSurf WITH DukTape    "
              echo "------------------------------------------------"
              echo "                                                "
              NETSURF_DUKTAPE="WithDukTape"
       else
              echo "                                                "
              echo "------------------------------------------------"
              echo " You chose to compile NetSurf WITHOUT DukTape  "
              echo "------------------------------------------------"
              echo "                                                "
              NETSURF_DUKTAPE="NoDukTape"
fi

NETSURF_ARCHIVE_NAME="NetSurf_$NETSURF_VERSION""_$NETSURF_DUKTAPE""_AmigaOS3.tar"

if [ "$keepFiles" = "Y" -o "$keepFiles" = "y" ]
       then
              echo "                                                "
              echo "------------------------------------------------"
              echo "       You chose to KEEP the files after        "
              echo "            the NetScript Execution             "
              echo "------------------------------------------------"
              echo "                                                "
       else
              echo "                                                "
              echo "------------------------------------------------"
              echo "      You chose to DELETE the files after       "
              echo "            the NetScript Execution             "
              echo "------------------------------------------------"
              echo "                                                "
fi

if [ "$keepOptNetSurf" = "Y" -o "$keepOptNetSurf" = "y" ]
       then
              echo "                                                "
              echo "------------------------------------------------"
              echo "   You chose to KEEP the /opt/netsurf folder    "
              echo "       at the end of NetScript execution        "
              echo "    for future shorter NetScript executions     "
              echo "------------------------------------------------"
              echo "                                                "
       else
              echo "                                                "
              echo "------------------------------------------------"
              echo "  You chose to DELETE the /opt/netsurf folder   "
              echo "       at the end of NetScript execution        "
              echo " (future NetScript executions will be longer)   "
              echo "------------------------------------------------"
              echo "                                                "
fi
```

```
	if [ "$cleanWorkspace" = "Y" -o "$cleanWorkspace" = "y" ]
		then
			echo "                                                "
			echo "------------------------------------------------"
			echo "       You chose to CLEAN your Workspace        "
			echo "        BEFORE the NetScript execution          "
			echo "      to compile a fresh NetSurf version        "
			echo " (your previous modifications will be deleted) "
			echo "------------------------------------------------"
			echo "                                                "

			echo "                                                "
			echo "_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>"
			echo "                  Cleaning                      "
			echo "_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>_/>"
			echo "                                                "

			echo "Deleting old NetScript files..."

			rm -Rf buildsystem libcss libdom libhubbub libnsbmp libnsgif libnsutils
libparserutils libsvgtiny libutf8proc libwapcaplet nsgenbind toolchains NetSurf_*_AmigaOS3.tar
			rm -Rf netsurf/*
			rm -Rf netsurf

			echo "Old NetScript files deleted !"

		else
			echo "                                                "
			echo "------------------------------------------------"
			echo "   You chose to KEEP your Workspace as it is    "
			echo "        BEFORE the NetScript execution          "
			echo "      to compile a custom NetSurf version       "
			echo "         containing your modifications          "
			echo "  (your previous modifications will be saved)   "
			echo "------------------------------------------------"
			echo "                                                "
			rm -Rf NetSurf_*_AmigaOS3.tar
	fi

	echo "                                                  "
	echo "<><><><><><><><><><><><><><><><><><><><><><><><><>"
	echo "                 Cygwin Tools                     "
	echo "<><><><><><><><><><><><><><><><><><><><><><><><><>"
	echo "                                                  "

	echo "Installing Cygwin tools..."

	lynx -source rawgit.com/transcode-open/apt-cyg/master/apt-cyg > apt-cyg
	install apt-cyg /bin
	apt-cyg install wget git make patch pkg-config apache2 python subversion gperf flex
bison autoconf gcc-g++ gcc-tools-epoch2-automake libidn-devel kde-dev-scripts
	rm apt-cyg

	echo "Cygwin tools installed !"
```

```
	echo "                                                  "
	echo "+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+"
	echo "                   Toolchains                     "
	echo "+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+"
	echo "                                                  "

	echo "Verifying /opt/netsurf folder... "
	if [ ! -d "/opt/netsurf" ]
	then
		echo "/opt/netsurf folder doesn't exist."
		echo "NetScript is now going to download and compile the toolchains needed to create /opt/netsurf (the NetScript execution will be longer). "
		git clone git://git.netsurf-browser.org/toolchains.git
		./updateFiles.sh toolchains &
		cd toolchains/m68k-unknown-amigaos
		unlink /usr/bin/autom4te2.64
		ln -s /opt/gcc-tools/epoch2/bin/autom4te-2.64 /usr/bin/autom4te2.64
		make distclean
		make
		cd ../sdk
		make		GCCSDK_INSTALL_CROSSBIN=/opt/netsurf/m68k-unknown-amigaos/cross/bin GCCSDK_INSTALL_ENV=/opt/netsurf/m68k-unknown-amigaos/env distclean
		make		GCCSDK_INSTALL_CROSSBIN=/opt/netsurf/m68k-unknown-amigaos/cross/bin GCCSDK_INSTALL_ENV=/opt/netsurf/m68k-unknown-amigaos/env
		cd ../..

		mkdir -p /opt/netsurf/m68k/unknown/amigaos/cross/bin/m68k/unknown/amigaos
		cp		/opt/netsurf/m68k-unknown-amigaos/cross/m68k-unknown-amigaos/bin/* /opt/netsurf/m68k/unknown/amigaos/cross/bin/m68k/unknown/amigaos

		cp	/opt/netsurf/m68k/unknown/amigaos/cross/bin/m68k/unknown/amigaos/ar.exe /opt/netsurf/m68k/unknown/amigaos/cross/bin/m68k/unknown/amigaos/-ar.exe
		rm /opt/netsurf/m68k/unknown/amigaos/cross/bin/m68k/unknown/amigaos/ar.exe
	else
		echo "/opt/netsurf folder already exists."
		echo "NetScript doesn't need to download and compile the toolchains (the NetScript execution will be shorter). "
	fi

		export PATH=/opt/netsurf/m68k-unknown-amigaos/cross/bin:$PATH
		export		PKG_CONFIG_PATH=/opt/netsurf/m68k-unknown-amigaos/env/lib/pkgconfig:$PKG_CONFIG_PATH
```

```
	echo "                                                    "
	echo "$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$"
	echo "                  Build System                      "
	echo "$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$;$"
	echo "                                                    "

	echo "Verifying buildsystem folder... "
	if [ ! -d "buildsystem" ]
	then
		echo "buildsystem folder doesn't exist. Downloading the fresh one..."
		git clone git://git.netsurf-browser.org/buildsystem.git
		cd buildsystem
		git pull
		make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-
unknown-amigaos clean
	else
		echo "buildsystem folder already exists. Compiling with your modifications (if you
did some)..."
		cd buildsystem
	fi

	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos
	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos install
	cd ..

	echo "                                                    "
	echo "/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/"
	echo "                    Libnsgif                        "
	echo "/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/"
	echo "                                                    "

	echo "Verifying libnsgif folder... "
	if [ ! -d "libnsgif" ]
	then
		echo "libnsgif folder doesn't exist. Downloading the fresh one..."
		git clone git://git.netsurf-browser.org/libnsgif.git
		cd libnsgif
		git pull
		make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-
unknown-amigaos clean
	else
		echo "libnsgif folder already exists. Compiling with your modifications (if you
did some)..."
		cd libnsgif
	fi

	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos
	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos install
	cd ..
```

```
	echo "                                                    "
	echo "^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^"
	echo "                    Libnsbmp                        "
	echo "^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^"
	echo "                                                    "

	echo "Verifying libnsbmp folder... "
	if [ ! -d "libnsbmp" ]
	then
		echo "libnsbmp folder doesn't exist. Downloading the fresh one..."
		git clone git://git.netsurf-browser.org/libnsbmp.git
		cd libnsbmp
		git pull
		make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-
unknown-amigaos clean
	else
		echo "libnsbmp folder already exists. Compiling with your modifications (if you
did some)..."
		cd libnsbmp
	fi

	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos
	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos install
	cd ..

	echo "                                                    "
	echo "%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%"
	echo "                   Libwapcaplet                     "
	echo "%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%|%"
	echo "                                                    "

	echo "Verifying libwapcaplet folder... "
	if [ ! -d "libwapcaplet" ]
	then
		echo "libwapcaplet folder doesn't exist. Downloading the fresh one..."
		git clone git://git.netsurf-browser.org/libwapcaplet.git
		cd libwapcaplet
		git pull
		make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-
unknown-amigaos clean
	else
		echo "libwapcaplet folder already exists. Compiling with your modifications (if
you did some)..."
		cd libwapcaplet
	fi

	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos
	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos install
	cd ..
```

```
	echo "                                                    "
	echo "?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?"
	echo "                  Libparserutils                    "
	echo "?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?!?"
	echo "                                                    "

	echo "Verifying libparserutils folder... "
	if [ ! -d "libparserutils" ]
	then
		echo "libparserutils folder doesn't exist. Downloading the fresh one..."
		git clone git://git.netsurf-browser.org/libparserutils.git
		cd libparserutils
		git pull
		make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-
unknown-amigaos clean
	else
		echo "libparserutils folder already exists. Compiling with your modifications (if
you did some)..."
		cd libparserutils
	fi

	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos
	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos install
	cd ..

	echo "                                                    "
	echo "0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o"
	echo "                    Libhubbub                       "
	echo "0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o"
	echo "                                                    "

	echo "Verifying libhubbub folder... "
	if [ ! -d "libhubbub" ]
	then
		echo "libhubbub folder doesn't exist. Downloading the fresh one..."
		git clone git://git.netsurf-browser.org/libhubbub.git
		cd libhubbub
		git pull
		rm -Rf examples
		rm -Rf perf
		make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-
unknown-amigaos clean
	else
		echo "libhubbub folder already exists. Compiling with your modifications (if you
did some)..."
		cd libhubbub
	fi

	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos
	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos install
	cd ..
```

```
	echo "                                                    "
	echo "#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#"
	echo "                      Libdom                        "
	echo "#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#:#"
	echo "                                                    "

	echo "Verifying libdom folder... "
	if [ ! -d "libdom" ]
	then
		echo "libdom folder doesn't exist. Downloading the fresh one..."
		git clone git://git.netsurf-browser.org/libdom.git
		cd libdom
		git pull
		rm -Rf examples
		make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-
unknown-amigaos clean
	else
		echo "libdom folder already exists. Compiling with your modifications (if you did
some)..."
		cd libdom
	fi

	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos
	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos install
	cd ..

	echo "                                                    "
	echo "*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*"
	echo "                      Libcss                        "
	echo "*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*"
	echo "                                                    "

	echo "Verifying libcss folder... "
	if [ ! -d "libcss" ]
	then
		echo "libcss folder doesn't exist. Downloading the fresh one..."
		git clone git://git.netsurf-browser.org/libcss.git
		cd libcss
		git pull
		make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-
unknown-amigaos clean
	else
		echo "libcss folder already exists. Compiling with your modifications (if you did
some)..."
		cd libcss
	fi

	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos
	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos install
	cd ..
```

```
	echo "                                                  "
	echo "_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_"
	echo "                    Libsvgtiny                    "
	echo "_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_&_"
	echo "                                                  "

	echo "Verifying libsvgtiny folder... "
	if [ ! -d "libsvgtiny" ]
	then
		echo "libsvgtiny folder doesn't exist. Downloading the fresh one..."
		git clone git://git.netsurf-browser.org/libsvgtiny.git
		cd libsvgtiny
		git pull
		make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-
unknown-amigaos clean
	else
		echo "libsvgtiny folder already exists. Compiling with your modifications (if you
did some)..."
		cd libsvgtiny
	fi

	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos
	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos install
	cd ..

	echo "                                                  "
	echo "@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@"
	echo "                    Libnsutils                    "
	echo "@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@¨@"
	echo "                                                  "

	echo "Verifying libnsutils folder... "
	if [ ! -d "libnsutils" ]
	then
		echo "libnsutils folder doesn't exist. Downloading the fresh one..."
		git clone git://git.netsurf-browser.org/libnsutils.git
		cd libnsutils
		git pull
		make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-
unknown-amigaos clean
	else
		echo "libnsutils folder already exists. Compiling with your modifications (if you
did some)..."
		cd libnsutils
	fi

	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos
	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos install
	cd ..
```

```
	echo "                                                    "
	echo "=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*="
	echo "                   Libutf8proc                      "
	echo "=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*="
	echo "                                                    "

	echo "Verifying libutf8proc folder... "
	if [ ! -d "libutf8proc" ]
	then
		echo "libutf8proc folder doesn't exist. Downloading the fresh one..."
		git clone git://git.netsurf-browser.org/libutf8proc.git
		cd libutf8proc
		git pull
		make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-
unknown-amigaos clean
	else
		echo "libutf8proc folder already exists. Compiling with your modifications (if you
did some)..."
		cd libutf8proc
	fi

	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos
	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   HOST=m68k-unknown-
amigaos install
	cd ..

	echo "                                                    "
	echo "{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}"
	echo "                     Nsgenbind                      "
	echo "{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}~{=}"
	echo "                                                    "

	echo "Verifying nsgenbind folder... "
	if [ ! -d "nsgenbind" ]
	then
		echo "nsgenbind folder doesn't exist. Downloading the fresh one..."
		git clone git://git.netsurf-browser.org/nsgenbind.git
		cd nsgenbind
		git pull
		make PREFIX=/opt/netsurf/m68k-unknown-amigaos/env clean
	else
		echo "nsgenbind folder already exists. Compiling with your modifications (if you
did some)..."
		cd nsgenbind
	fi

	make PREFIX=/opt/netsurf/m68k-unknown-amigaos/env
	make PREFIX=/opt/netsurf/m68k-unknown-amigaos/env install
	cd ..

	unlink /usr/bin/nsgenbind
	ln    -s    $(PWD)/nsgenbind/build-i686-pc-cygwin-i686-pc-cygwin-release-binary/nsgenbind
/usr/bin/nsgenbind
```

```
	echo "                                                  "
	echo "~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~"
	echo "                    NetSurf                       "
	echo "~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~"
	echo "                                                  "

	echo "Verifying netsurf folder... "
	if [ ! -d "netsurf" ]
	then
		echo "netsurf folder doesn't exist. Downloading the fresh one..."
		git clone git://git.netsurf-browser.org/netsurf.git
		./updateFiles.sh amiga &
		cd netsurf
		git pull
		make TARGET=amigaos3 PREFIX=/opt/netsurf/m68k-unknown-amigaos/env CC=m68k-unknown-
amigaos-gcc clean
	else
		echo "netsurf folder already exists. Compiling with your modifications (if you did
some)..."
		cd netsurf
	fi

	if [ "$compileWithDukTape" = "Y" -o "$compileWithDukTape" = "y" ]
		then
			echo "                                                  "
			echo "--------------------------------------------------"
			echo "                    REMINDER                      "
			echo "   You chose to compile NetSurf WITH DukTape      "
			echo "--------------------------------------------------"
			echo "                                                  "
			echo override NETSURF_USE_DUKTAPE := YES >> Makefile.config.example
			cp Makefile.config.example Makefile.config
		else
			echo "                                                  "
			echo "--------------------------------------------------"
			echo "                    REMINDER                      "
			echo "  You chose to compile NetSurf WITHOUT DukTape    "
			echo "--------------------------------------------------"
			echo "                                                  "
			echo override NETSURF_USE_DUKTAPE := NO >> Makefile.config.example
			cp Makefile.config.example Makefile.config
	fi

	make   TARGET=amigaos3   PREFIX=/opt/netsurf/m68k-unknown-amigaos/env   CC=m68k-unknown-
amigaos-gcc package
	mv NetSurf_Amiga/netsurf.tar ../$NETSURF_ARCHIVE_NAME
	cd ..
```

```
	if [ "$keepFiles" = "Y" -o "$keepFiles" = "y" ]
		then
			echo "                                                "
			echo "------------------------------------------------"
			echo "                    REMINDER                    "
			echo "       You chose to KEEP the files after        "
			echo "            the NetScript Execution             "
			echo "------------------------------------------------"
			echo "                                                "
		else
			echo "                                                "
			echo "------------------------------------------------"
			echo "                    REMINDER                    "
			echo "      You chose to DELETE the files after       "
			echo "            the NetScript Execution             "
			echo "------------------------------------------------"
			echo "                                                "
			rm -Rf buildsystem libcss libdom libhubbub libnsbmp libnsgif libnsutils
libparserutils libsvgtiny libutf8proc libwapcaplet nsgenbind toolchains
			rm -Rf netsurf/*
			rm -Rf netsurf
	fi

	if [ "$keepOptNetSurf" = "Y" -o "$keepOptNetSurf" = "y" ]
		then
			echo "                                                "
			echo "------------------------------------------------"
			echo "                    REMINDER                    "
			echo "   You chose to KEEP the /opt/netsurf folder    "
			echo "       at the end of NetScript execution        "
			echo "    for future shorter NetScript executions     "
			echo "------------------------------------------------"
			echo "                                                "
		else
			echo "                                                "
			echo "------------------------------------------------"
			echo "                    REMINDER                    "
			echo "  You chose to DELETE the /opt/netsurf folder   "
			echo "       at the end of NetScript execution        "
			echo " (future NetScript executions will be longer)   "
			echo "------------------------------------------------"
			echo "                                                "
			rm -Rf /opt/netsurf
	fi
```

```
	if [ -f "$NETSURF_ARCHIVE_NAME" ]
		then
			echo "                                                "
			echo "------------------------------------------------"
			echo "                  GOOD ENDING                   "
			echo "     Your NetSurf Archive is available in :     "
			echo "     $(PWD)"
			echo "                And is called :                 "
			echo "     $NETSURF_ARCHIVE_NAME"
			echo "                                                "
			echo "     Unpack it into a folder reachable by       "
			echo "      your AmigaOS3 and then install it.        "
			echo "(LHA Archive Format is not supported by Cygwin,"
			echo "         sorry for the inconvenience)           "
			echo "                                                "
			echo "         Thanks for using NetScript !           "
			echo "             Spread the world ! ^^              "
			echo "------------------------------------------------"
			echo "                                                "
		else
			echo "                                                "
			echo "------------------------------------------------"
			echo "                   BAD ENDING                   "
			echo "    The NetSurf Archive hasn't been created.    "
			echo "       That means an issue has been done        "
			echo "          during the compilation. :-(           "
			echo "                                                "
			echo "      Please try to launch NetScript again      "
			echo "    in order to compile the toolchains again    "
			echo "      by deleting the /opt/netsurf folder       "
			echo "      and cleaning your Workspace folder        "
			echo "         (this could solve the issue).          "
			echo "                                                "
			echo "You can also send to DNADNL the log file called"
			echo "               LOG_NetScript.txt                "
			echo "and located in $(PWD)"
			echo "           via the Amiga.org forums.            "
			echo "    DNADNL will do his best to answer you.      "
			echo "                                                "
			echo "         Sorry for the inconvenience !          "
			echo "------------------------------------------------"
			echo "                                                "

			while [ \( "$deleteOptNetSurf" != "Y" -a "$deleteOptNetSurf" != "N" \) -a
\( "$deleteOptNetSurf" != "y" -a "$deleteOptNetSurf" != "n" \) ]
                        do read -p "(1/2) Do you want to DELETE the /opt/netsurf folder now
in order to recompile it (this could solve the issue encountered) ? This question is asked even
if you already deleted it, in that case, NetScript can't make the folder reappear. (Y/n) : "
deleteOptNetSurf
			done
```

```
		while [ \( "$deleteFiles" != "Y" -a "$deleteFiles" != "N" \) -a \( "$deleteFiles" != "y" -a "$deleteFiles" != "n" \) ]
			do read -p "(2/2) Do you want to DELETE the files and libraries used for the compilation ? This question is asked even if you already deleted them, in that case, NetScript can't make the files and libraries reappear. (Y/n) : " deleteFiles
		done

		if [ "$deleteOptNetSurf" = "Y" -o "$deleteOptNetSurf" = "y" ]
			then
				echo "                                                "
				echo "------------------------------------------------"
				echo "  You chose to DELETE the /opt/netsurf folder   "
				echo "------------------------------------------------"
				echo "                                                "
				rm -Rf /opt/netsurf
				echo "                                                "
				echo "------------------------------------------------"
				echo "          /opt/netsurf folder deleted !         "
				echo "------------------------------------------------"
				echo "                                                "
			else
				echo "                                                "
				echo "------------------------------------------------"
				echo "   You chose to KEEP the /opt/netsurf folder    "
				echo "------------------------------------------------"
				echo "                                                "
		fi

		if [ "$deleteFiles" = "Y" -o "$deleteFiles" = "y" ]
			then
				echo "                                                "
				echo "------------------------------------------------"
				echo "  You chose to DELETE the files and libraries   "
				echo "------------------------------------------------"
				echo "                                                "
				rm -Rf buildsystem libcss libdom libhubbub libnsbmp libnsgif libnsutils libparserutils libsvgtiny libutf8proc libwapcaplet nsgenbind toolchains
				rm -Rf netsurf/*
				rm -Rf netsurf
				echo "                                                "
				echo "------------------------------------------------"
				echo "          Files and libraries deleted !         "
				echo "------------------------------------------------"
				echo "                                                "
			else
				echo "                                                "
				echo "------------------------------------------------"
				echo "   You chose to KEEP the files and libraries    "
				echo "------------------------------------------------"
				echo "                                                "
		fi
```

```
                echo "                                                "
                echo "------------------------------------------------"
                echo "        Now, you can relaunch NetScript         "
                echo "    by using the \"./NetScript.sh\" command     "
                echo "     (sorry again for the inconvenience !)      "
                echo "                                                "
                echo "         Thanks for using NetScript !           "
                echo "             Spread the world ! ^^              "
                echo "------------------------------------------------"
                echo "                                                "
        fi


else

        echo "                                                "
        echo "------------------------------------------------"
        echo "         NetScript for Netsurf $NETSURF_VERSION"
        echo "Crossed Compilation Cygwin (Windows) - AmigaOS3"
        echo "                                                "
        echo "       Created by DNADNL, EyMenZ & Tygre        "
        echo "                                                "
        echo "               A big thanks to :                "
        echo "          Chris Young, transcode-open           "
        echo "           and the amiga.org forums !           "
        echo "                                                "
        echo "     *** Script Version : $NETSCRIPT_DATE ***   "
        echo "------------------------------------------------"
        echo "                                                "

        echo "                                                "
        echo "------------------------------------------------"
        echo "             ~~~~ QUICK MODE ~~~~               "
        echo "     This version only recompiles NetSurf       "
        echo "     You need to launch NetScript normally      "
        echo "    at least once to use this quick version     "
        echo "------------------------------------------------"
        echo "                                                "

        while [ \( "$compileWithDukTape" != "Y" -a "$compileWithDukTape" != "N" \) -a \(
"$compileWithDukTape" != "y" -a "$compileWithDukTape" != "n" \) ]
                do read -p "(1/1) Do you want to compile NetSurf WITH DukTape ?(Y/n) : "
compileWithDukTape
        done
```

```
	if [ "$compileWithDukTape" = "Y" -o "$compileWithDukTape" = "y" ]
		then
			echo "                                                "
			echo "------------------------------------------------"
			echo "   You chose to compile NetSurf WITH DukTape    "
			echo "------------------------------------------------"
			echo "                                                "
			NETSURF_DUKTAPE="WithDukTape"
		else
			echo "                                                "
			echo "------------------------------------------------"
			echo " You chose to compile NetSurf WITHOUT DukTape   "
			echo "------------------------------------------------"
			echo "                                                "
			NETSURF_DUKTAPE="NoDukTape"
	fi

	NETSURF_ARCHIVE_NAME="NetSurf_$NETSURF_VERSION""_$NETSURF_DUKTAPE""_AmigaOS3.tar"

	rm -Rf NetSurf_*_AmigaOS3.tar

	export PATH=/opt/netsurf/m68k-unknown-amigaos/cross/bin:$PATH
	export                                    PKG_CONFIG_PATH=/opt/netsurf/m68k-unknown-
amigaos/env/lib/pkgconfig:$PKG_CONFIG_PATH
	unlink /usr/bin/nsgenbind
	ln    -s    $(PWD)/nsgenbind/build-i686-pc-cygwin-i686-pc-cygwin-release-binary/nsgenbind
/usr/bin/nsgenbind

	echo "                                                  "
	echo "~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~"
	echo "                    NetSurf                       "
	echo "~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~'~.~"
	echo "                                                  "

	cd netsurf

	if [ "$compileWithDukTape" = "Y" -o "$compileWithDukTape" = "y" ]
		then
			echo "                                                "
			echo "------------------------------------------------"
			echo "                    REMINDER                    "
			echo "   You chose to compile NetSurf WITH DukTape    "
			echo "------------------------------------------------"
			echo "                                                "
			echo override NETSURF_USE_DUKTAPE := YES >> Makefile.config.example
			cp Makefile.config.example Makefile.config
		else
			echo "                                                "
			echo "------------------------------------------------"
			echo "                     REMINDER                   "
			echo "  You chose to compile NetSurf WITHOUT DukTape  "
			echo "------------------------------------------------"
			echo "                                                "
			echo override NETSURF_USE_DUKTAPE := NO >> Makefile.config.example
			cp Makefile.config.example Makefile.config
	fi
```

```
	make	TARGET=amigaos3	PREFIX=/opt/netsurf/m68k-unknown-amigaos/env	CC=m68k-unknown-
amigaos-gcc package
	mv NetSurf_Amiga/netsurf.tar ../$NETSURF_ARCHIVE_NAME
	cd ..

	if [ -f "$NETSURF_ARCHIVE_NAME" ]
		then
			echo "                                                "
			echo "------------------------------------------------"
			echo "                  GOOD ENDING                   "
			echo "     Your NetSurf Archive is available in :     "
			echo "     $(PWD)"
			echo "                And is called :                 "
			echo "     $NETSURF_ARCHIVE_NAME"
			echo "                                                "
			echo "     Unpack it into a folder reachable by       "
			echo "      your AmigaOS3 and then install it.        "
			echo "(LHA Archive Format is not supported by Cygwin,"
			echo "         sorry for the inconvenience)           "
			echo "                                                "
			echo "         Thanks for using NetScript !           "
			echo "             Spread the world ! ^^              "
			echo "------------------------------------------------"
			echo "                                                "
		else
			echo "                                                "
			echo "------------------------------------------------"
			echo "                  BAD ENDING                    "
			echo "    The NetSurf Archive hasn't been created.    "
			echo "       That means an issue has been done        "
			echo "          during the compilation. :-(           "
			echo "                                                "
			echo "      Please try to launch NetScript again      "
			echo "    by using the \"./NetScript.sh\" command     "
			echo "    in order to compile the toolchains again    "
			echo "      by deleting the /opt/netsurf folder       "
			echo "      and cleaning your Workspace folder        "
			echo "         (this could solve the issue).          "
			echo "                                                "
			echo "You can also send to DNADNL the log file called"
			echo "               LOG_NetScript.txt                "
			echo "and located in $(PWD)"
			echo "           via the Amiga.org forums.            "
			echo "    DNADNL will do his best to answer you.      "
			echo "                                                "
			echo "         Sorry for the inconvenience !          "
			echo "                                                "
			echo "         Thanks for using NetScript !           "
			echo "             Spread the world ! ^^              "
			echo "------------------------------------------------"
			echo "                                                "
	fi
fi
```

12.3.2 - NetScript.sh

```
if [ "$1" = "-q" -o "$1" = "--quick" ]
	then
		./NS.sh -q 2>&1 | tee LOG_NetScript.txt
	else
		./NS.sh 2>&1 | tee LOG_NetScript.txt
fi
```

12.3.3 - updateFiles.sh

```
if [ $1 = "toolchains" ]
then
	echo -en "Searching : m68k-unknown-amigaos/Makefile "

	while [ ! -f "toolchains/m68k-unknown-amigaos/Makefile" ]
	do
		echo -en "."
		sleep 0.5
	done
	echo " m68k-unknown-amigaos/Makefile found !"

	sed -f updateFile/updateFile_m68k-unknown-amigaos_Makefile.sed toolchains/m68k-unknown-amigaos/Makefile > toolchains/m68k-unknown-amigaos/MakefileNEW
	cp toolchains/m68k-unknown-amigaos/MakefileNEW toolchains/m68k-unknown-amigaos/Makefile
	rm toolchains/m68k-unknown-amigaos/MakefileNEW

	echo -en "Searching : sdk/Makefile "

	while [ ! -f "toolchains/sdk/Makefile" ]
	do
		echo -en "."
		sleep 0.5
	done
	echo " sdk/Makefile found !"

	sed -f updateFile/updateFile_sdk_Makefile.sed toolchains/sdk/Makefile > toolchains/sdk/MakefileNEW
	cp toolchains/sdk/MakefileNEW toolchains/sdk/Makefile
	rm toolchains/sdk/MakefileNEW

	echo -en "Searching : c-parse.in "

	while [ ! -f "toolchains/m68k-unknown-amigaos/gcc-3.4.6/gcc/c-parse.in" ]
	do
		echo -en "."
		sleep 0.5
	done
	echo " c-parse.in found !"

	sed -f updateFile/updateFile_c-parse.in.sed toolchains/m68k-unknown-amigaos/gcc-3.4.6/gcc/c-parse.in > toolchains/m68k-unknown-amigaos/gcc-3.4.6/gcc/c-parseNEW.in
	cp toolchains/m68k-unknown-amigaos/gcc-3.4.6/gcc/c-parseNEW.in toolchains/m68k-unknown-amigaos/gcc-3.4.6/gcc/c-parse.in
	rm toolchains/m68k-unknown-amigaos/gcc-3.4.6/gcc/c-parseNEW.in
```

```
	echo -en "Searching : collect2.c "

	while [ ! -f "toolchains/m68k-unknown-amigaos/gcc-3.4.6/gcc/collect2.c" ]
	do
		echo -en "."
		sleep 0.5
	done
	echo " collect2.c found !"

	sed	-f	updateFile/updateFile_collect2.c.sed	toolchains/m68k-unknown-amigaos/gcc-3.4.6/gcc/collect2.c > toolchains/m68k-unknown-amigaos/gcc-3.4.6/gcc/collect2NEW.c
	cp  toolchains/m68k-unknown-amigaos/gcc-3.4.6/gcc/collect2NEW.c  toolchains/m68k-unknown-amigaos/gcc-3.4.6/gcc/collect2.c
	rm toolchains/m68k-unknown-amigaos/gcc-3.4.6/gcc/collect2NEW.c

elif [ $1 = "amiga" ]
then
	echo -en "Searching : frontends/amiga/Makefile "

	while [ ! -f "netsurf/frontends/amiga/Makefile" ]
	do
		echo -en "."
		sleep 0.5
	done
	echo " frontends/amiga/Makefile found !"

	sed		-f		updateFile/updateFile_frontends_amiga_Makefile.sed netsurf/frontends/amiga/Makefile > netsurf/frontends/amiga/MakefileNEW
	cp netsurf/frontends/amiga/MakefileNEW netsurf/frontends/amiga/Makefile
	rm netsurf/frontends/amiga/MakefileNEW

else
	echo "Please specify an argument : toolchains or amiga."
fi

exit
```

12.3.4 - Files from the updateFile folder

updateFile_collect2.c.sed

```
s/ redir_handle = open (redir, O_WRONLY | O_TRUNC | O_CREAT);/ redir_handle = open (redir, O_WRONLY | O_TRUNC | O_CREAT, 0666);/g
```

updateFile_c-parse.in.sed

```
s/YYLEX/yylex()/g
```

updateFile_frontends_amiga_Makefile.sed

```
s#lha a netsurf.lha NetSurf NetSurf.info AutoInstall#tar -cvf netsurf.tar NetSurf NetSurf.info AutoInstall#g
s#AMIGA_LANGUAGES := de en it ja nl#AMIGA_LANGUAGES := de en it ja nl fr#g
s#lha#tar#g
```

updateFile_m68k-unknown-amigaos_Makefile.sed

```
s/UPSTREAM_GUIGFX_TARBALL := guigfxlib.lha/UPSTREAM_GUIGFX_TARBALL := guigfxlib/g
s#UPSTREAM_GUIGFX_URI := http://neoscientists.org/~bifat/binarydistillery/$(UPSTREAM_GUIGFX_TARBALL)#UPSTREAM_GUIGFX_URI := https://github.com/DNADNL/guigfxlib.git#g
s/UPSTREAM_RENDER_TARBALL := renderlib.lha/UPSTREAM_RENDER_TARBALL := renderlib/g
s#UPSTREAM_RENDER_URI := http://neoscientists.org/~bifat/binarydistillery/$(UPSTREAM_RENDER_TARBALL)#UPSTREAM_RENDER_URI := https://github.com/DNADNL/renderlib.git#g
s#mkdir -p $(BUILDDIR)/guigfxlib#cp -R $(UPSTREAM_GUIGFX_TARBALL) $(BUILDDIR)#g
s#lha xw=$(BUILDDIR)/guigfxlib $(SOURCESDIR)/$(UPSTREAM_GUIGFX_TARBALL)#rm -R $(UPSTREAM_GUIGFX_TARBALL)#g
s#mkdir -p $(BUILDDIR)/renderlib#cp -R $(UPSTREAM_RENDER_TARBALL) $(BUILDDIR)#g
s#lha xw=$(BUILDDIR)/renderlib $(SOURCESDIR)/$(UPSTREAM_RENDER_TARBALL)#rm -R $(UPSTREAM_RENDER_TARBALL)#g
s#renderlib/renderlib/include/#renderlib/include/#g
s#wget -q -O $@ $(UPSTREAM_GUIGFX_URI)#git clone $(UPSTREAM_GUIGFX_URI)#g
s#wget -q -O $@ $(UPSTREAM_RENDER_URI)#git clone $(UPSTREAM_RENDER_URI)#g
```

updateFile_sdk_Makefile.sed

```
s#cc__ := $(word 1,$(wildcard $(GCCSDK_INSTALL_CROSSBIN)/\*gcc))#cc__ := $(GCCSDK_INSTALL_CROSSBIN)/m68k-unknown-amigaos-gcc#g
s#cxx__ := $(word 1,$(wildcard $(GCCSDK_INSTALL_CROSSBIN)/\*g++))#cxx__ := $(GCCSDK_INSTALL_CROSSBIN)/m68k-unknown-amigaos-g++#g
s#ar__ := $(word 1,$(wildcard $(GCCSDK_INSTALL_CROSSBIN)/\*ar))#ar__ := $(GCCSDK_INSTALL_CROSSBIN)/m68k-unknown-amigaos-ar#g
s#ranlib__ := $(word 1,$(wildcard $(GCCSDK_INSTALL_CROSSBIN)/\*ranlib))#ranlib__ := $(GCCSDK_INSTALL_CROSSBIN)/m68k-unknown-amigaos-ranlib#g
```

12.4 - The DukTape Issue AmiKit LOG file

```
duk_js_compiler.c:416 duk__advance_helper: duk__advance_helper() BEGIN
duk_js_compiler.c:417 duk__advance_helper: duk__advance_helper() : duk_hthread *thr = comp_ctx->thr;
duk_js_compiler.c:419 duk__advance_helper: duk__advance_helper() : duk_context *ctx = (duk_context *) thr;
duk_js_compiler.c:421 duk__advance_helper: duk__advance_helper() : duk_bool_t regexp;
duk_js_compiler.c:424 duk__advance_helper: duk__advance_helper() : DUK_ASSERT(comp_ctx->curr_token.t >= 0 && comp_ctx->curr_token.t <= DUK_TOK_MAXVAL);
duk_js_compiler.c:436 duk__advance_helper: duk__advance_helper() : regexp = 1;
duk_js_compiler.c:438 duk__advance_helper: duk__advance_helper() : if (duk__token_lbp[comp_ctx->curr_token.t] & DUK__TOKEN_LBP_FLAG_NO_REGEXP) BEGIN
duk_js_compiler.c:440 duk__advance_helper: duk__advance_helper() : if (duk__token_lbp[comp_ctx->curr_token.t] & DUK__TOKEN_LBP_FLAG_NO_REGEXP) : regexp = 0;
duk_js_compiler.c:443 duk__advance_helper: duk__advance_helper() : if (duk__token_lbp[comp_ctx->curr_token.t] & DUK__TOKEN_LBP_FLAG_NO_REGEXP) END
duk_js_compiler.c:444 duk__advance_helper: duk__advance_helper() : if (comp_ctx->curr_func.reject_regexp_in_adv) BEGIN
duk_js_compiler.c:451 duk__advance_helper: duk__advance_helper() : if (comp_ctx->curr_func.reject_regexp_in_adv) END
duk_js_compiler.c:453 duk__advance_helper: duk__advance_helper() : if (expect >= 0 && comp_ctx->curr_token.t != expect) BEGIN
duk_js_compiler.c:455 duk__advance_helper: duk__advance_helper() : if (expect >= 0 && comp_ctx->curr_token.t != expect) : DUK_D(DUK_DPRINT(parse error: expect=315438081, got=290994208, (long) expect, (long) comp_ctx->curr_token.t));
duk_js_compiler.c:458 duk__advance_helper: duk__advance_helper() : if (expect >= 0 && comp_ctx->curr_token.t != expect) : DUK_ERROR_SYNTAX(thr, DUK_STR_PARSE_ERROR);
duk_error_longjmp.c:9 duk_err_longjmp: duk_err_longjmp() BEGIN
duk_error_longjmp.c:10 duk_err_longjmp: duk_err_longjmp() : DUK_ASSERT(thr != NULL);
duk_error_longjmp.c:13 duk_err_longjmp: duk_err_longjmp() : DUK_DD(DUK_DDPRINT(longjmp error: type=290994208 iserror=315402414 value1=!T value2=!T, (int) thr->heap->lj.type, (int) thr->heap->lj.iserror, &thr->heap->lj.value1, &thr->heap->lj.value2));
duk_error_longjmp.c:27 duk_err_longjmp: duk_err_longjmp() : if (!thr->heap->lj.jmpbuf_ptr) BEGIN
duk_error_longjmp.c:40 duk_err_longjmp: duk_err_longjmp() : if (!thr->heap->lj.jmpbuf_ptr) END
duk_error_longjmp.c:51 duk_err_longjmp: duk_err_longjmp() : DUK_LONGJMP(thr->heap->lj.jmpbuf_ptr->jb);
duk_api_call.c:222 duk_safe_call: rc = duk_handle_safe_call(thr, func, nargs, nrets); OK ! et rc = 1
duk_js_compiler.c:8322 duk_js_compile: duk_js_compile() : safe_rc = 1
duk_js_compiler.c:8323 duk_js_compile: duk_js_compile() : thr->compile_ctx = prev_ctx;
duk_js_compiler.c:8326 duk_js_compile: duk_js_compile() : if (safe_rc != DUK_EXEC_SUCCESS) BEGIN
duk_js_compiler.c:8328 duk_js_compile: duk_js_compile() : if (safe_rc != DUK_EXEC_SUCCESS) : duk_throw(ctx);
duk_api_stack.c:4309 duk_throw: duk_throw() BEGIN
duk_api_stack.c:4310 duk_throw: duk_throw() : duk_hthread *thr = (duk_hthread *) ctx;
duk_api_stack.c:4313 duk_throw: duk_throw() : DUK_ASSERT(thr->valstack_bottom >= thr->valstack);
duk_api_stack.c:4315 duk_throw: duk_throw() : DUK_ASSERT(thr->valstack_top >= thr->valstack_bottom);
duk_api_stack.c:4317 duk_throw: duk_throw() : DUK_ASSERT(thr->valstack_end >= thr->valstack_top);
```

```
duk_api_stack.c:4320 duk_throw: duk_throw() : if (thr->valstack_top == thr->valstack_bottom)
BEGIN
duk_api_stack.c:4325 duk_throw: duk_throw() : if (thr->valstack_top == thr->valstack_bottom)
END
duk_api_stack.c:4336 duk_throw: duk_throw() : duk_hthread_sync_and_null_currpc(thr);
duk_api_stack.c:4340 duk_throw: duk_throw() : DUK_DDD(DUK_DDDPRINT(THROW ERROR (API): !dT
(before throw augment), (duk_tval *) duk_get_tval(ctx, -1)));
duk_api_stack.c:4342 duk_throw: duk_throw() : duk_err_augment_error_throw(thr);
duk_api_stack.c:4345 duk_throw: duk_throw() : DUK_DDD(DUK_DDDPRINT(THROW ERROR (API): !dT
(after throw augment), (duk_tval *) duk_get_tval(ctx, -1)));
duk_api_stack.c:4348 duk_throw: duk_throw() : duk_err_setup_heap_ljstate(thr,
DUK_LJ_TYPE_THROW);
duk_api_stack.c:4356 duk_throw: duk_throw() : duk_err_longjmp(thr);
duk_error_longjmp.c:9 duk_err_longjmp: duk_err_longjmp() BEGIN
duk_error_longjmp.c:10 duk_err_longjmp: duk_err_longjmp() : DUK_ASSERT(thr != NULL);
duk_error_longjmp.c:13 duk_err_longjmp: duk_err_longjmp() : DUK_DD(DUK_DDPRINT(longjmp error:
type=290994208 iserror=315475192 value1=!T value2=!T, (int) thr->heap->lj.type, (int) thr-
>heap->lj.iserror, &thr->heap->lj.value1, &thr->heap->lj.value2));
duk_error_longjmp.c:27 duk_err_longjmp: duk_err_longjmp() : if (!thr->heap->lj.jmpbuf_ptr)
BEGIN
duk_error_longjmp.c:30 duk_err_longjmp: duk_err_longjmp() : if (!thr->heap->lj.jmpbuf_ptr) :
DUK_D(DUK_DPRINT(uncaught error: type=290994208 iserror=315475192 value1=!T value2=!T, (int)
thr->heap->lj.type, (int) thr->heap->lj.iserror, &thr->heap->lj.value1,
 &thr->heap->lj.value2));
duk_error_longjmp.c:35 duk_err_longjmp: duk_err_longjmp() : if (!thr->heap->lj.jmpbuf_ptr) :
duk_fatal((duk_context *) thr, DUK_ERR_UNCAUGHT_ERROR, uncaught error);
duk_api_stack.c:4364 duk_fatal: duk_fatal() BEGIN
duk_api_stack.c:4365 duk_fatal: duk_fatal() : duk_hthread *thr = (duk_hthread *) ctx;
duk_api_stack.c:4368 duk_fatal: duk_fatal() : DUK_ASSERT_CTX_VALID(ctx);
duk_api_stack.c:4370 duk_fatal: duk_fatal() : DUK_ASSERT(thr != NULL);
duk_api_stack.c:4372 duk_fatal: duk_fatal() : DUK_ASSERT(thr->heap != NULL);
duk_api_stack.c:4374 duk_fatal: duk_fatal() : DUK_ASSERT(thr->heap->fatal_func != NULL);
duk_api_stack.c:4377 duk_fatal: duk_fatal() : DUK_D(DUK_DPRINT(fatal error occurred, code
290994208, message Nq/
$o, (long) err_code, (const char *) err_msg));
duk_api_stack.c:4385 duk_fatal: duk_fatal() : thr->heap->fatal_func(ctx, err_code, err_msg);
FATAL 56: uncaught error
PANIC 56: uncaught error (calling abort)
Abnormal program termination
NetSurf : erreur code 20
```

Abstract

During my 4th year in Computer Science and Management at Polytech MONTPELLIER, I had the opportunity to do an internship with Yann-Gaël GUEHENEUC (Ptidej Team leader) at École Polytechnique de Montréal, Canada. The project consisted in porting / compiling an Internet browser to an old but still alive version of an operating system called Amiga OS 3.X (aka AmigaOS3), released in 1992. There, I could draw my inspiration from a previous student work on an Internet browser called NetSurf. First, I had to seek if another browser could be ported to AmigaOS3 (TimberWolf, Odyssey...). Unfortunately, because of technical limitations, no browser is a reliable competitor of NetSurf. Then, I developed NetScript, a Shell script to automate the NetSurf compilation for AmigaOS3 on Windows and which helps the NetSurf community to get involved in its development (to add features, to debug it...). Finally, I had to track NetSurf bugs and then import it on Happiga, an Amiga distribution for the Raspberry Pi micro-computer.

Key Words:

Amiga, AmigaOS3, Browser, Community, Compilation, Computer Science, Cygwin, Debugging, École Polytechnique de Montréal, Import, Internet, Internship, NetScript, NetSurf, Operating System, Ptidej Team, Polytech Montpellier, Port, Portage, Raspberry Pi, Research, Shell, Software Engineering, Windows