

Une erreur est survenue lors du chargement de la version complète de ce site. Veuillez vider le cache de votre navigateur et rafraîchir cette page pour corriger cette erreur.

## Interface de compilation/reboot/logging avancé pour les développeurs Core

Galathil

**Important :** Ce tutoriel a été rédigé par Nobody pour le site wow-emu.fr. Je le recopie ici à l'identique, Il se peut que certaines informations soient obsolètes à ce jour. Bonne lecture !

# Interface de compilation/reboot/logging avancé pour les développeurs Core

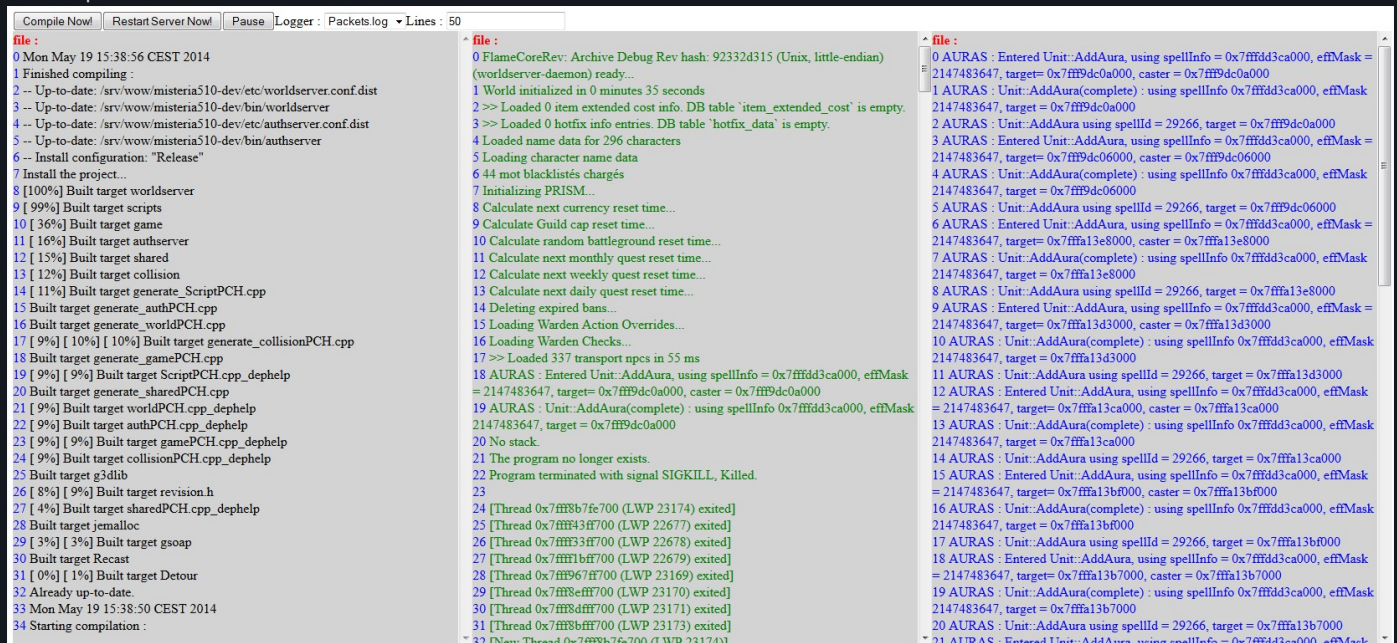
Bonjour à tous,

aujourd'hui, je vous propose d'utiliser le WebCompiler, une interface que j'avais développé il y a quelques mois pour mon serveur Mist-Eria.

Fonctionnalités disponibles

- Interface Web regroupant trois pages de log différentes.
  - Page 1 : Compilation. Affiche le résultat de la dernière compilation effectuée. Affiche en rouge les lignes comportant une erreur (pour une meilleure identification des problèmes de compilation).
  - Page 2 : Crash. Affiche les derniers crashes du serveur avec un rapport GDB de l'erreur, ainsi qu'un bout (20 lignes) du fichier de log principal afin d'avoir éventuellement des informations supplémentaires (spellid qui fait crash, par exemple).
  - Page 3 : Logs. Affiche une page de log spécifique.
- L'ensemble de ces pages sont dynamiques. Elles sont mises à jour toutes les secondes. Nul besoin de faire F5 pour mettre à jour l'affichage.
- Plusieurs personne peuvent voir l'interface et l'utiliser en même temps.
- Il y a également quatre fonctionnalités en haut à gauche qui permettent de gérer le serveur.
  - Le bouton de recompilation. Lance un script qui effectue un git pull puis une recompilation du serveur.
  - Le bouton de redémarrage. Redémarre le serveur de jeu (en réalité, le fait crasher, puis le redémarrer le redémarre juste après).
  - Le bouton de pause. Permet de mettre en Pause la mise à jour de l'affichage des trois fenêtres (pour pouvoir copier coller un message d'erreur, par exemple).
  - Le champ de sélection du log à afficher et le nombre de lignes affichées. Permet de choisir le nombre de lignes et le log affiché sur la fenêtre de droite.

Voici à quoi ressemble l'interface :



```
file :
0 Mon May 19 15:38:56 CEST 2014
1 Finished compiling :
2 -- Up-to-date: /srv/wow/mistéria510-dev/etc/worldserver.conf.dist
3 -- Up-to-date: /srv/wow/mistéria510-dev/bin/worldserver
4 -- Up-to-date: /srv/wow/mistéria510-dev/etc/authserver.conf.dist
5 -- Up-to-date: /srv/wow/mistéria510-dev/bin/authserver
6 -- Install configuration: "Release"
7 Install the project...
8 [100%] Built target worldserver
9 [ 99%] Built target scripts
10 [ 36%] Built target game
11 [ 16%] Built target authserver
12 [ 15%] Built target shared
13 [ 12%] Built target collision
14 [ 11%] Built target generate_ScriptPCH.cpp
15 Built target generate_authPCH.cpp
16 Built target generate_worldPCH.cpp
17 [ 0%] [ 10%] [ 10%] Built target generate_collisionPCH.cpp
18 Built target generate_gamePCH.cpp
19 [ 0%] [ 9%] Built target ScriptPCH.cpp_dephelp
20 Built target generate_sharedPCH.cpp
21 [ 9%] Built target worldPCH.cpp_dephelp
22 [ 9%] Built target authPCH.cpp_dephelp
23 [ 9%] [ 9%] Built target gamePCH.cpp_dephelp
24 [ 9%] Built target collisionPCH.cpp_dephelp
25 Built target g3dlib
26 [ 8%] [ 9%] Built target revision.h
27 [ 4%] Built target sharedPCH.cpp_dephelp
28 Built target jemalloc
29 [ 3%] [ 3%] Built target gsoap
30 Built target Recast
31 [ 0%] [ 1%] Built target Detour
32 Already up-to-date.
33 Mon May 19 15:38:50 CEST 2014
34 Starting compilation :

file :
0 FlameCoreRev: Archive Debug Rev hash: 92332d315 (Unix, little-endian)
(worldserver-daemon) ready...
1 World initialized in 0 minutes 35 seconds
2 >> Loaded 0 item extended cost info. DB table 'item_extended_cost' is empty.
3 >> Loaded 0 hotfix info entries. DB table 'hotfix_data' is empty.
4 Loaded name data for 296 characters
5 Loading character name data
6 44 mot blacklistés chargés
7 Initializing PRISM...
8 Calculate next currency reset time...
9 Calculate Guild cap reset time...
10 Calculate random battleground reset time...
11 Calculate next monthly quest reset time...
12 Calculate next weekly quest reset time...
13 Calculate next daily quest reset time...
14 Deleting expired bans...
15 Loading Warden Action Overrides...
16 Loading Warden Checks...
17 --> Loaded 337 transport npc's in 55 ms
18 AURAS : Entered Unit::AddAura, using spellInfo = 0x7ffdd3ca000, effMask = 2147483647, target= 0x7ff9dc0a000, caster = 0x7ff9dc0a000
19 AURAS : Unit::AddAura(complete) : using spellInfo 0x7ffdd3ca000, effMask 2147483647, target= 0x7ff9dc0a000
20 No stack.
21 The program no longer exists.
22 Program terminated with signal SIGKILL, Killed.
23
24 [Thread 0x7ff8b7fe700 (LWP 23174) exited]
25 [Thread 0x7ff8b43ff00 (LWP 22677) exited]
26 [Thread 0x7ff8b3ff700 (LWP 22678) exited]
27 [Thread 0x7ff8b1bff00 (LWP 22679) exited]
28 [Thread 0x7ff967ff700 (LWP 23169) exited]
29 [Thread 0x7ff8effff700 (LWP 23170) exited]
30 [Thread 0x7ff8dffff700 (LWP 23171) exited]
31 [Thread 0x7ff8bfbff700 (LWP 23173) exited]
32 [New Thread 0x7ff8b7fe700 (LWP 23174)]

file :
0 AURAS : Entered Unit::AddAura, using spellInfo = 0x7ffdd3ca000, effMask = 2147483647, target= 0x7ff9dc0a000, caster = 0x7ff9dc0a000
1 AURAS : Unit::AddAura(complete) : using spellInfo 0x7ffdd3ca000, effMask 2147483647, target= 0x7ff9dc0a000
2 AURAS : Unit::AddAura using spellId = 29266, target = 0x7ff9dc0a000
3 AURAS : Entered Unit::AddAura, using spellInfo = 0x7ffdd3ca000, effMask = 2147483647, target= 0x7ff9dc06000, caster = 0x7ff9dc06000
4 AURAS : Unit::AddAura(complete) : using spellInfo 0x7ffdd3ca000, effMask 2147483647, target = 0x7ff9dc06000
5 AURAS : Unit::AddAura using spellId = 29266, target = 0x7ff9dc06000
6 AURAS : Entered Unit::AddAura, using spellInfo = 0x7ffdd3ca000, effMask = 2147483647, target= 0x7ffa13e8000, caster = 0x7ffa13e8000
7 AURAS : Unit::AddAura(complete) : using spellInfo 0x7ffdd3ca000, effMask 2147483647, target = 0x7ffa13e8000
8 AURAS : Unit::AddAura using spellId = 29266, target = 0x7ffa13e8000
9 AURAS : Entered Unit::AddAura, using spellInfo = 0x7ffdd3ca000, effMask = 2147483647, target= 0x7ffa13d3000, caster = 0x7ffa13d3000
10 AURAS : Unit::AddAura(complete) : using spellInfo 0x7ffdd3ca000, effMask 2147483647, target = 0x7ffa13d3000
11 AURAS : Unit::AddAura using spellId = 29266, target = 0x7ffa13d3000
12 AURAS : Entered Unit::AddAura, using spellInfo = 0x7ffdd3ca000, effMask = 2147483647, target= 0x7ffa13b7000, caster = 0x7ffa13b7000
13 AURAS : Unit::AddAura(complete) : using spellInfo 0x7ffdd3ca000, effMask 2147483647, target = 0x7ffa13b7000
14 AURAS : Unit::AddAura using spellId = 29266, target = 0x7ffa13b7000
15 AURAS : Entered Unit::AddAura, using spellInfo = 0x7ffdd3ca000, effMask = 2147483647, target= 0x7ffa13bf000, caster = 0x7ffa13bf000
16 AURAS : Unit::AddAura(complete) : using spellInfo 0x7ffdd3ca000, effMask 2147483647, target = 0x7ffa13bf000
17 AURAS : Unit::AddAura using spellId = 29266, target = 0x7ffa13bf000
18 AURAS : Entered Unit::AddAura, using spellInfo = 0x7ffdd3ca000, effMask = 2147483647, target= 0x7ffa13b7000, caster = 0x7ffa13b7000
19 AURAS : Unit::AddAura(complete) : using spellInfo 0x7ffdd3ca000, effMask 2147483647, target = 0x7ffa13b7000
20 AURAS : Unit::AddAura using spellId = 29266, target = 0x7ffa13b7000
21 AURAS : Entered Unit::AddAura, using spellInfo = 0x7ffdd3ca000, effMask = 2147483647, target= 0x7ffa13b7000, caster = 0x7ffa13b7000
```

Le revers de la médaille, c'est que ces différentes fonctionnalités demandent d'être configurées correctement, et c'est un peu long. C'est donc l'objet de la seconde partie.

# Configuration du système

## 1) Pré-requis

Ce système, pour être totalement fonctionnel, à besoin de plusieurs applications pour fonctionner.

Premièrement, le système utilise plusieurs scripts BASH. Il vous faut donc être sous Linux.

Deuxièmement, le système a besoin de faire tourner des scripts en continu. Il vous faut donc (pour simplifier le travail) utiliser le logiciel screen. Je ne donnerais pas d'explications sur son fonctionnement ici, sachez juste que ça permet de garder des consoles ouvertes même lorsque vous vous déconnectez du serveur.

Troisièmement, le système a besoin d'un serveur Web avec le support du PHP. Moi, j'utilise Apache, mais libre à vous d'utiliser le serveur que vous souhaitez.

Quatrièmement, le système utilise gdb pour récupérer les crashes du serveur. La encore, je ne fournirais pas d'explications supplémentaires. Pour gdb, il n'y aura "rien" à comprendre, seulement à compiler votre serveur en mode Debug (option WITH\_COREDEBUG à 1) et utiliser le script que je mets à votre disposition.

Enfin, le système utilise le système de log interne du serveur. Cela veut dire que vous devez avoir compris le fonctionnement des Loggers et Appenders dans le fichier worldserver.conf du serveur.

Ces scripts sont donc à destination d'un public un minimum habitué à Linux, les débutants devront malheureusement passer leur chemin.

## 2) configuration des logs

La première chose à faire est de configurer votre serveur pour générer des logs utilisables. Je n'expliquerais pas ici comment fonctionnent les Appenders et Loggers, la documentation dans le worldserver.conf est suffisante. La seule chose nécessaire est de configurer le logsDir pour que les logs soient sauvegardés dans un sous dossier. Pour ma part, ce sera le répertoire logs (donc serverdir/bin/logs/).

## 3) configuration du restarter

Premièrement, placez vous dans le dossier de votre jeu (le dossier bin) et créez le dossier webcompiler (donc `serverdir/bin/webcompiler/` ).

```
mkdir webcompiler
```

puis donnez lui des accès 777 afin que notre utilisateur web puisse y avoir accès. Donnez également ces mêmes accès au dossier logs pour les mêmes raisons.

```
chmod -R 777 webcompiler
chmod -R 777 logs
```

Configurez également le pidFile dans le worldserver.conf pour avoir un fichier contenant le pid (numéro de processus) du serveur. Cela permettra de le redémarrer par la suite.

Ensuite, copiez et collez ce code dans le fichier restarter.sh (donc `serverdir/bin/restarter.sh` ) :

```
#!/bin/sh

crashlog="logs/Crash.log"
serverlog="logs/Server.log"
while true; do
  gdb -batch -x gdb.sh worldserver
  sleep 2
  tail -20 $serverlog >> gdb.txt
  cp gdb.txt $crashlog
```

```
cp gdb.txt $crashlog
done
```

Les deux variables crashlog et serverlog sont configurables si nécessaires (crashlog sera le fichier de log contenant les crashes, serverlog est le fichier de log général généré par le serveur).

Le second script à créer est le script gdb.sh (donc `serverdir/bin/gdb.sh`) :

```
set logging on
set pagination off

run
bt full

quit
```

Vous pouvez modifier le nom du script, mais dans ce cas il faut modifier son appel dans le `restarter.sh`

Donnez ensuite le droit d'exécution à ces deux fichiers

```
chmod +x restarter.sh gdb.sh
```

Enfin, créez une nouvelle fenêtre screen et exécutez le `restarter.sh`

Voilà, c'est tout pour la partie `restarter`

## 4) Configuration du Compiler

je pars du principe que vous avez un dossier de compilation dans lequel vous avez créé le dossier build dans lequel vous compilez votre serveur.

Déplacez-vous dans ce dossier build et créez les deux fichiers suivants :

`compile.sh` :

```
git pull
make -j7 install
```

Si votre serveur n'est pas lié à un dépôt git tiers sur lequel vous envoyez vos modifications, enlevez simplement la première ligne (celle avec le `git pull`).

`autocompile.sh` :

```
#!/bin/sh

base="/serverdir/bin/";
datadir=$base"webcompiler/"
logsdir=$base"logs/"
pidFile=$base"world.pid"

compile=$datadir"compile"
restart=$datadir"restart"
outputCompile=$logsdir"compile.log"

while true; do
  if [ -e $restart ]; then
    echo "Restarting server"
    kill -9 `cat $pidFile`
```

```

kill -9 $(cat $pidfile)
sleep 5
rm $restart
elif [ -e $compile ]; then
echo "Starting compilation"
echo "Starting compilation :" > $outputCompile
date >> $outputCompile
./compile.sh 1>>$outputCompile 2>>$outputCompile
echo "Finished compiling"
echo "Finished compiling :" >> $outputCompile
date >> $outputCompile
sleep 5
rm $compile
else
echo "Nothing to compile"
sleep 5
fi
done

```

Les quatre premières variables sont à configurer. Il faut définir le répertoire de base de votre serveur (là où est exécuté le worldserver), le répertoire webcompiler créé en 3), le dossier de logs, et enfin le nom du fichier contenant le pid, également défini dans le worldserver.conf.

Une fois encore, ajoutez le droit d'exécution à ces deux scripts.

```

chmod +x compile.sh autocompile.sh

```

Enfin, exécutez le fichier autocompile.sh dans un second screen afin qu'il tourne en boucle.

Voilà qui est fait pour le compiler. Il nous reste encore l'interface web.

## 5) L'interface web

Dernier élément, mais non des moindres, l'interface web qui va relier tout ça ☐

Premièrement, il faut récupérer les fichiers du dossier web contenu dans l'archive mise à votre disposition et les coller dans un dossier de votre serveur web. Pour moi ce sera le dossier webcompiler.

Deuxièmement, il vous faut ouvrir le fichier config.php

et modifier le fichier en conséquence (adaptez le chemin vers le dossier de votre serveur. Si vous souhaitez afficher d'autres logs que le Server.log, il vous suffit d'ajouter le nom du fichier de log en reprenant le même schéma, après avoir modifié votre worldserver.conf pour générer ce fichier évidemment).

Il ne vous reste plus qu'à vous rendre dans votre navigateur, à taper l'adresse de cette interface, et d'apprécier les logs qui défilent devant vos yeux ébahis :3

Derniers rappels au cas où

Comme la procédure d'installation est assez longue, je rappelle les éléments principaux :

L'archive comporte 3 dossiers :

- Web contient tous les fichiers à mettre dans l'interface web. Il faut ensuite modifier le config.php pour adapter l'interface à votre serveur.
- restart contient les 2 fichiers nécessaires au redémarrage de votre serveur. Ces fichiers sont à mettre à côté de votre worldserver. N'oubliez pas de créer le dossier webcompiler de modifier le restart.sh puis le worldserver.conf pour ajouter le pid et enregistrer les logs dans le sous-dossier logs.
- compiler contient les 2 fichiers nécessaires à la recompilation et au redémarrage de votre serveur. N'oubliez pas de modifier le fichier autocompile.sh pour qu'il soit également adapté à votre serveur.

J'ai tout dit il me semble, face à la complexité de la procédure, j'ai demandé à plusieurs personnes de tester chez eux pour voir si cela est réalisable,

Comme toujours, je reste disponible pour toute demande supplémentaire

# Téléchargement

L'archive est disponible [ici](#)

---