

DailyBuild

第 1.0 版 – 2017/3/13

本ドキュメントでは、Springhead の日々のメンテナンス作業 (dailybuild の結果の確認、Wiki ページのメンテナンス等) およびそれらに関わるスクリプトについて説明する。

目次

1	DailyBuild とは	2
2	ルーチン作業	3
3	Wiki のメンテナンス	4
4	Visual Studio のバージョン	5
5	タスク	6
5.1	タスク TestMain	6
5.2	タスク MakeReport	6
5.3	タスク実行結果の参照	6
6	スクリプトの説明	7
6.1	TestMain.bat	7
6.1.1	TestAll.bat	9
6.1.2	BuildVC.bat	12
6.2	MakeReport.bat	15
6.2.1	build_monitor.bat	17
6.2.2	base_lib.pm	20
6.2.3	dailybuild_lib.pm	21

1 DailyBuild とは

Springhead2 に加えられた修正または新規に組み込まれたソースと既存ソースとの整合性の確認、また何らかの理由によるソースの劣化の防止等を目的として、次の処理を Windows タスクとして日々 (自動的に) 実行している。

1. Repository から最新バージョンを取り出し、著作権の関係で公開できないソースを削除した後 Stub および Springhead2 Library を作成する。
2. tests/Samples 以下にあるテストソリューションに対してビルド&テストを実行する。現在は Visual Studio 2013 を使用し、Win32/Debug の構成でビルドを行っている。
3. その結果 (ログファイル) を Repository へ登録する。
4. ログファイル、最新ドキュメン (Doxygen で作成)、使用したソースおよび公開用 tarball を Web に登録して公開する。

この一連のタスクを DailyBuild と呼ぶ。

また、DailyBuild の結果を Wiki に反映させるためのタスクも日々実行している。このタスクを MakeReport と呼ぶ。

2 ルーチン作業

DailyBuild の結果を Wiki (<http://springhead.info/wiki/index.php?devel>) で確認する。

1. 次の 2 つのリンクを辿り、以前の結果と相違がないか確認する。
 - 開発版 - 今朝のビルド - 今朝のビルドの状況 (過去のビルドの履歴)
 - 開発版 - 今朝のビルド - Samples のビルドの状況 (過去のビルドの履歴)
2. 相違があったときは「今朝のビルドのログ」等を見て原因を特定する。
 - テスト系の不具合のときは修正をする。
 - 開発系の不具合のときは開発者に連絡をし、必要なら対処を依頼する。
 - メール先 : springhead 開発者 <dev@springhead.info>
 - さらにバグトラッカへ登録も検討する (最近は活用されていない)。
 - リンク : Wiki 開発者向け情報のページにある (ポップアップ画面に注意)
 - 今朝のビルドが最後まで実行できていない場合には、DailyBuild を実行するマシン MediaServer(192.168.91.4) の
 - D:\DailyBuild\Springhead2\test\dailybuild.log
 - D:\DailyBuild\Springhead2\test\makereport.logを見て原因を特定する必要がある。
3. 週に 1 回程度は次も確認する。
 - (a) ダウンロードページから開発版 tar ファイルをダウンロードし、内容を確認する。
 - (b) ドキュメントページから API リファレンス のリンクを辿り、ページが正しく作成されているか確認する。特に、ヘッダファイルのインクルード依存関係図が生成されているか確認する。(例えば、[ユーティリティクラス]-[BaseUtility.h] と辿ってみる)
 - (c) ドキュメントページから Windows Help ファイルをダウンロードし、上記と同様に内容を確認する。ダウンロードしたファイル ("Springhead.chm") は、プロパティウィンドウを開き、セキュリティフィールドの「ブロックの解除」を行なわないと内容が表示されないの注意すること。

DailyBuild の実行に関わるスクリプトについては、“6 スクリプトの説明”を参照のこと。

3 Wiki のメンテナンス

Wiki ページ (<http://springhead.info/wiki/>) のうち、

Springhead ダウンロード <http://springhead.info/wiki/download>

Springhead ドキュメント <http://springhead.info/wiki/document>

Springhead 開発者向け情報 <http://springhead.info/wiki/devel>

のメンテナンスを行なう (他のページも必要があれば)。

今朝のビルドにある「テーブル」及び「過去のビルドの履歴」を作成するスクリプトについては、ドキュメント “Wiki ページのメンテナンス” を参照のこと。

4 Visual Studio のバージョン

現在サポートしている Visual studio は、

Visual Studio 2015 (Platform Toolset v140)

Visual Studio 2013 (Platform Toolset v120)

Visual Studio 2010 (Platform Toolset v100)

である。現在の主たるメンテナンス対象バージョンは Visual Studio 2013 であるが、DailyBuild はこれらのいずれのバージョンでも実行できるようにメンテナンスする必要がある (他のバージョンおよび他の構成のビルド&テストは DailyBuild とは別途行なう必要がある)。

他に Visual Studio 2008 と Visual Studio 2012 のソリューションファイルもあるが、これらは最新の状態にはメンテナンスされていない。今後もメンテナンスする必要はない。

特に次の場合に注意すること。

1. ヘッドファイルまたはソースファイルが追加・削除された場合

プロジェクトファイルへの反映が、開発者が使用している Visual Studio のバージョンのみに限られてコミットされていることが多いで、すべてが同じとなるようプロジェクトファイルをメンテナンスする必要がある。

[参考] フォルダにある "DependCheckAll.bat" を src ディレクトリで実行すると依存ファイルの差分が分かる。ただし完全ではないかもしれない。

2. 新しいテストが追加された場合

src/tests または src/Samples の下に新しいディレクトリが作られてそこに追加された場合には、基本的には自動で組み込まれる。ただし、ディレクトリ名とソリューション名とが異なるケースなどではうまく働かないので対処が必要である。

“6.1.2 BuildVC.bat” を参照のこと。

3. Wiki の「今朝のビルド」欄にある「レビジョン nnnn (yyyy/mm/dd) との比較」の比較の対象となるレビジョンは、

"D:\DailyBuild\Springhead2\test\Minitoring\etc\revision.old"

"D:\DailyBuild\Springhead2\test\Minitoring\etc\revision.new"

の二つのファイルで指定されている。これらの比較は、テスト実行が出力するテキストの相違から機能・動作の変化を捉えるための手掛かりとして実施している。

“6.2 MakeReport.bat” の「関連ファイル」を参照のこと。

5 タスク

DailyBuild は毎日タスクとして実行される。

実行マシン MediaServer (192.168.91.4)

タスク名 Task1: TestMain 毎日 02:03 に起動

 Task2: MakeReport 毎日 06:03 に起動

5.1 タスク TestMain

ビルド&テストを実行するタスク

Script "D:\DailyBuild\Springhead2\test\TestMain.bat"

作業領域 "D:\DailyBuild\Springhead2Test\"

概要 作業領域に最新バージョンを Checkout してそこでライブラリ作成および
 ビルド&テストを実行

結果 "D:\DailyBuild\Springhead2Test\test\log\" に作成

詳細 “6.1 TestMain.bat” を参照のこと。

5.2 タスク MakeReport

過去のレビジョンでの実行結果との差分情報を作成するタスク

Script "D:\DailyBuild\Springhead2\test\MakeReport.bat"

概要 指定された過去レビジョンの実行結果と最新レビジョンの実行結果とを比較
 較して差分情報を作成し、レポートファイルにまとめる。

 過去レビジョンの指定はファイル

 "D:\DailyBuild\Springhead2\test\Monitoring\etc\revision.old"
 で行なう。

詳細 “6.2 MakeReport.bat” を参照のこと。

5.3 タスク実行結果の参照

ウェブ springhead.info/wiki の “開発者向け情報” から参照できる。

TestMain → ‘今朝のビルドの状況’,
 → ‘Samples のビルドの状況’,
 → ‘今朝のビルドのログ’

MakeReport → ‘今朝のビルドについてまとめた報告’,
 → ‘レビジョン nnnn (yyyy/mm/dd) との比較’

6 スクリプトの説明

タスク DailyBuild およびタスク MakeReport を実行するためのスクリプトについて説明する。

6.1 TestMain.bat

タスク TestMain を実行するためのスクリプト (Windows batch file)。DailyBuild 全体 (MakeReport は含まない) の制御を行なう。DailyBuild で行なう作業については “1 DailyBuild とは” を参照のこと。

引数：

/t toolset_id	ツールセットの識別 {10.0 11.0 12.0 <default> ..}
/c config	ビルド構成 { Debug <default> Release .. }
/p platform	ビルドプラットフォーム { Win32 <default> x64 }
/y python_ver	Python のバージョン { 32 33 34 <default> }
/r test_rep	テスト用ローカルコピーのルート <default: Springhead2Test>
/h	使用方法の表示

関連スクリプト：

```
D:¥DailyBuild¥
├── Springhead2¥
│   ├── test¥
│   │   ├── TestMain.bat ... DailyBuild 全体の制御 ... 本スクリプト
│   │   └── bat¥
│   │       ├── TestAll.bat ... ビルド&テストの制御 ... 6.1.1 参照)
│   │       ├── BuildVC.bat ... 個々のビルド&テストの実行 ... 6.1.2 参照)
│   │       ├── SwigAll.bat ... Swig の実行
│   │       └── MakeDoc.bat ... ドキュメント作成の制御
│   ├── src¥
│   │   ├── DelAll.bat ... 不要となったファイルの削除
│   │   └── MakePack.bat ... 公開 tarball の作成
```

処理を制御する環境変数：

以下に示す処理の説明の各ステップは、実行時の環境変数で制御することができる。各ステップの「制御変数名」に記述された環境変数が "skip" と設定されているときに限

りそのステップは実行されない。毎日のタスクにおける DailyBuild の実行時にはこれらの環境変数はすべて未定義としておく。

これはデバッグ及びローカルでのテストのときに使用する。

ディレクトリ構成：

DailyBuild は、"D:\DailyBuild\Springhead2\test"を現在ディレクトリとして起動するが、スクリプト内部では相対ディレクトリを使用しているので、以下では絶対パス形式表示をしているのに関わらず、"D:\DailyBuild\Springhead2"というドライブ・ディレクトリ名には意味はない。ただし、引数 "/r test_rep" に指定するディレクトリは、スクリプト起動ディレクトリから 2 階層上のディレクトリ (前記の "D:\DailyBuild") からの相対指定となっている必要がある。

処理の説明：

1. Springhead2/test の更新を反映する

制御変数名 DAILYBUILD_UPDATE_SPRINGHEAD2

処理 "D:\DailyBuild\Springhead2\"以下を最新バージョンにする。

2. Springhead2Test にソースツリーを取得する

制御変数名 DAILYBUILD_CLEANUP_WORKSPACE

処理 テストディレクトリ (Springhead2Test – 引数/r で変更可) 以下に最新バージョンを Checkout する。

3. 公開できないファイルの削除と設定変更

制御変数名 DAILYBUILD_CLEANUP_WORKSPACE

処理 公開できないファイルを削除し、関連する設定を変更する。

4. テストを行なう

制御変数名 DAILYBUILD_EXECUTE_TESTALL

もう少し細かい制御については“6.1.1 TestAll.bat”を参照のこと。

処理 ビルド&テストを実行する ("TestAll.bat")。

呼出し時の引数は次のとおり

第 1 引数 ツールセット ID = 12.0 (引数 /t で変更可)

第 2 引数 ビルド構成 = Debug (引数 /c で変更可)

第 3 引数 プラットフォーム = Win32 (引数 /p で変更可)

第 4 引数 Python バージョン = 34 (引数 /y で変更可)

処理の詳細は“6.1.1 TestAll.bat”を参照のこと。

5. ドキュメントを作る (doxygen)

制御変数名 DAILYBUILD_EXECUTE_MAKEDOC

処理 doxygen によりドキュメントを作成する ("MakeDoc.bat")

6. ソースツリー (%TEST_REPOSITORY% 以下) を Web にコピー

制御変数名 DAILYBUILD_COPYTO_WEBBASE
処理 作業ファイル等公開不要のものを削除 ("DelAll.bat") し
Web に公開するファイル群をサーバにコピーする
コピー先 \\haselab\HomeDirs\WWW\ (次の行に続く)
docroots\springhead\daily_build 以下

7. 外部公開用のアーカイブを作る

制御変数名 DAILYBUILD_MAKE_ARCHIVE
処理 公開用のアーカイブ "Springhead2.tgz" を "MakePack.bat"
で作成してサーバにコピーする
コピー先 \\haselab\HomeDirs\WWW\ (次の行に続く)
docroots\springhead\daily_build\pack

6.1.1 TestAll.bat

スクリプト "TestMain.bat" から呼び出される。以下で示すターゲットのすべてに対して、引数で指定された条件に従ったソリューションのビルド&テストを制御する。

ターゲットの指定方法：

- リスト変数 TARGET_LIST に "Springhead2Test\src 以下のディレクトリ名を列挙する。ただし先頭のターゲット Stub は特別 (ライブラリ作成のためのターゲット) である。ここに列挙したターゲットディレクトリおよびその直接のサブディレクトリすべてをビルド&テストの対象とする。
- リスト変数 DO_BUILD_LIST および DO_RUN_LIST の対応する値 (yes 又は no) により、ビルドまたはテストを実行するか否かを制御できる。

現在の設定：

Stub ... ビルドのみ
test ... ビルドおよびテスト
Samples ... ビルドのみ

また、ビルド&テスト対象ディレクトリに次の名称のファイルを置くことで、個々のビルド&テスト対象単位で実行するか否かを制御できる (空ファイルでよい)。

"dailybuild.do.build" ... ビルドを実行する
"dailybuild.do.run" ... テストを実行する
"dailybuild.dont.build" ... ビルドを実行しない (実行指定より優先)
"dailybuild.dont.run" ... テストを実行しない (実行指定より優先)

- ターゲットを追加するときは、TARGET_LIST に追加するだけでなく、ログファイル名の設定も追加する必要がある。関連するリスト変数は、

BLD_SUCC_LOG_LIST ... ビルドログファイル名 (成功ターゲット)
 BLD_FAIL_LOG_LIST ... ビルドログファイル名 (失敗ターゲット)
 RUN_SUCC_LOG_LIST ... 実行ログファイル名 (成功ターゲット)
 RUN_FAIL_LOG_LIST ... 実行ログファイル名 (失敗ターゲット)

- 同様に、ターゲットを追加するときは、実行制御変数名の設定も追加する必要がある。設定するリスト変数名は DO_CONTROL_LIST である。現在の設定値は、

```
DO_CONTRIL_LIST=DAILYBUILD_EXECUTE_STUBBUILD ^
                DAILYBUILD_EXECUTE_BUILDRUN ^
                DAILYBUILD_EXECUTE_SAMPLEBUILD
```

であり、それぞれ Stub, tests, Samples の実行を制御する。

これらは見易くするためにリスト変数として設定するが、使用するときには配列変数のように扱う (インデックスは TARGET_LIST に設定されたターゲットの順番)。

サブルーチン “:list_to_array” でリスト変数から擬似配列変数にする。

引数 :

toolset_id	ツールセットの識別 {10.0 11.0 12.0 ..}
config	ビルド構成 {Debug Release ..}
platform	ビルドプラットフォーム {Win32 x64}
python_version	Python のバージョン {32 33 34}
test_repository	作業用ローカルコピーのルート

関連スクリプト :

```
D:¥DailyBuild¥
├─ Springhead2¥
│   └─ test¥
│       └─ bat¥
│           ├── TestAll.bat ... ビルド&テスト全体の制御 ... 本スクリプト
│           ├── SwigAll.bat ... Swig の実行
│           └─ BuildVC.bat ... Visual Studio の実行
```

処理の説明 :

1. 必要なパスの設定

"devenv.exe" のあるディレクトリを環境変数 PATH に追加する。Visual Studio 2015 までは、この設定でよいが、今後の Visual Studio のバージョンによっては設定を変える必要があるかもしれない (変数 TOOLSET_ID の値に注意)。

2. 実行制御変数

ビルドまたはテストの実行を制御する環境変数名を、配列変数 DO_CONTROLS に設定

する。使用する環境変数は次のとおり。

DAILYBUILD_EXECUTE_TESTALL	… このバッチファイル自体の呼出しを制御
DAILYBUILD_EXECUTE_STUBBUILD	… ターゲット Stub の実行を制御
DAILYBUILD_EXECUTE_BUILDRUN	… ターゲット tests の実行を制御
DAILYBUILD_EXECUTE_SAMPLEBUILD	… ターゲット Samples の実行を制御

3. ビルド&実行

ターゲット TARGET_LIST で指定した各ターゲットに対して次を実行する。

(a) ターゲットの識別

実行するターゲットを識別し、ビルドおよびテストを実行するか否かを決定する。

(b) 変数の初期化

"result.log" に書き出す情報を初期化する。

(c) ログファイルの初期化

ログファイルに日付とタイトルを書き出す。

(d) stub/library の作成 (TARGET_LIST の先頭要素 [=Stub] のとき)

"SwigAll.bat" を呼び出して library を作成する。

第 1 引数 … ツールセット ID (引数 toolset_id で指定された値)

第 2 引数 … ビルド構成 (引数 configuration で指定された値)

第 3 引数 … プラットフォーム (引数 platform で指定された値)

(e) ビルド&実行 (TARGET_LIST の先頭要素以外の場合)

ターゲットで指定されたディレクトリ及びその直下のすべてのサブディレクトリについて、ソリューションファイル ("*[toolset_id].sln") が存在したならば、バッチファイル "BuildVC.bat" を呼び出してビルドを実行する。

第 1 引数 … ツールセット ID (引数 toolset_id で指定された値)

第 2 引数 … 現在のターゲット名

第 3 引数 … ソリューション名

第 4 引数 … ソリューションディレクトリ名

第 5 引数 … ビルド構成 (引数 configuration で指定された値)

第 6 引数 … プラットフォーム (引数 platform で指定された値)

第 7 引数 … ビルド指定 (yes or no)

第 8 引数 … テスト実行指定 (yes or no)

第 9 引数 … ログファイル名 (ビルド結果)

第 10 引数 … ログファイル名 (ビルドエラー)

第 11 引数 … ログファイル名 (実行結果)

第 12 引数 … ログファイル名 (実行エラー)

第 13 引数 … Python バージョン (引数 python_version で指定された値)

処理の詳細は “6.1.2 BuildVC.bat” を参照のこと。

ビルドに成功したらタグ登録の予約をする (AT_LEAST_ONE_BLD_SUCC ← 1)。

最後に処理の結果を "result.log" に書き出す。

"result.log" に書き出す順序 (ビルド成功 ビルド失敗 実行成功
実行失敗) 及び (3b) での初期化情報を変更すると、ウェブページが正しく
表示されなくなる恐れがあるので注意すること。

4. タグを登録

制御変数名	DAILYBUILD_COPYTO_TAGS
処理	タグ登録が予約されていたら (%AT_LEAST_ONE_BLD_SUCC%=1) ならば、サブバージョンにタグの登録を行なう。 次に Springhead2 の更新履歴を "History.log" に出力する。

5. ログを Samba にコピーする

制御変数名	DAILYBUILD_COPYTO_BUILDLOG
処理	ログファイルをサーバにコピーする
コピー先	\\haselab\HomeDirs\WWW\ (次の行に続く) docroots\springhead\daily_build 以下

6. ログを SVN にコミットする

制御変数名	DAILYBUILD_COMMIT_BUILDLOG
処理	ログファイルをサブバージョンにコミットする。 メッセージは “Autobuild done.” である。

6.1.2 BuildVC.bat

スクリプト "TestAll.bat" から呼び出される。引数で指定されたソリューションに対して、指定された条件に従ってビルド (およびテスト) を実行し、その結果をログファイルに出力する。

引数 :

toolset_id	ツールセットの識別 {10.0 11.0 12.0 ..}
testset	テストセット名 (表示用)
category	テストカテゴリー名 (表示用)
solution_dir	ソリューション名 (ディレクトリ名)
build_conf	ビルドオプション (構成 {Debug Release ..})
buld_plat	ビルドオプション (プラットフォーム {Win32 x64})
do_build	ビルド指定 {yes no}
do_run	実行指定 {yes no}
log_1	ビルド結果のログファイル名 (追記形式)
log_2	ビルドエラーのログファイル名 (追記形式)

log_3	実行結果のログファイル名 (追記形式)
log_4	実行エラーのログファイル名 (追記形式)
python_ver	Python のバージョン

呼び出し元との共有変数：

次の環境変数を使用して呼び出し元へ値 (処理結果) を返す。

BLD_SUCC	… ビルドに成功したら “ビルド成功”
BLD_FAIL	… ビルドに失敗したら “ビルド失敗”
BLD_SUCC_LIST	… ビルドに成功したソリューションのリスト
BLD_FAIL_LIST	… ビルドに失敗したソリューションのリスト
RUN_SUCC	… 実行に成功したら “実行成功”
RUN_FAIL	… 実行に失敗したら “実行失敗”
RUN_SUCC_LIST	… 実行に成功したソリューションのリスト
RUN_FAIL_LIST	… 実行に失敗したソリューションのリスト

関連ファイル：

"dailybuild.outdir"	… 出力ディレクトリの変更 2(b) ii 参照
"dailybuild.alias"	… ソリューションファイル名の変更 3b 参照

処理の説明：

1. パスの確認

変数 %VS_DIR% に対応するディレクトリがない場合はエラーとする。

今後の Visual Studio のバージョンによってはチェック方法を変える必要があるかもしれない。

2. 処理開始

(a) ファイル "dailybuild.{do|dont}.{build|run}" の存在により、引数 do_build, do_run での指定をオーバーライドする (変数 DO_BLD および DO_RUN で制御する)。

(b) 出力ディレクトリは次のようにして決定する。

i. 変数 OUTDIRSPEC のデフォルトを \$TOOLSET/\$PLATFORM/\$CONFIGURATION とする。

ii. 引数 solution_dir で指定されたディレクトリに次のファイルが存在したら、変数 OUTDIRSPEC の値をそのファイルの内容で置き換える。

ファイル： "dailybuild.outdir"

ただし引数 python_ver の値が 32 と 33 以降とでは処理が少々違うので注意 (これは 'Samples/EmbPython' のための仕組み)。

iii. 変数 OUTDIRSPEC 中の \$xx の部分をそれぞれ引数で指定された値に置き換え、それを出力ディレクトリとする。

3. ソリューションファイル名を決定する。

- (a) ソリューションファイル名のデフォルトは引数 `solution_dir` で指定された名称と同じとする。
- (b) 引数 `solution_dir` で指定されたディレクトリに次のファイルが存在したら、ソリューションファイル名はそのファイルの内容と同じとする。

ファイル: "dailybuild.alias"

4. ビルド

- (a) ビルドを実施するか否かは、これまでに変数 `DO_BLD` に設定された値に従う。
- (b) ビルドに成功したか否かは、所定のディレクトリに実行形式ファイルが生成されたか否かで判断する。
- (c) ビルドが成功した時は、環境変数 `BLD_SUCC` に“ビルド成功”を設定し、環境変数 `BLD_SUCC_LIST` にソリューション名が追加する。
ビルドが失敗した時は、環境変数 `BLD_FAIL` に“ビルド失敗”を設定し、環境変数 `BLD_FAIL_LIST` にソリューション名が追加する。

5. 実行

- (a) テスト実行を実施するか否かは、これまでに変数 `DO_RUN` に設定された値に従う。
- (b) テスト実行が成功したか否かは、環境変数 `ERRORLEVEL` に設定された値が 0 か否かに従う。
- (c) ビルドが成功した時は、環境変数 `RUN_SUCC` に“ビルド成功”を設定し、環境変数 `RUN_SUCC_LIST` にソリューション名が追加する。
ビルドが失敗した時は、環境変数 `RUN_FAIL` に“ビルド失敗”を設定し、環境変数 `RUN_FAIL_LIST` にソリューション名が追加する。

6.2 MakeReport.bat

タスク MakeReport を実行するためのスクリプト (Windows batch file)。DailyBuild の結果 (ログファイル) から、基準とするレビジョンの実行結果との差分情報をレポートファイルとして作成する。この情報は Wiki の “開発者向け情報” のページから参照できる。

URL: <http://springhead.info/wiki/index.php?devel>

[開発版] - [今朝のビルド] - [レビジョン ... との比較]

オプション引数:

-c	レポートファイルをウェブにコピーする
-k	作業ファイルを残す
-t	コピー元ディレクトリ指定 デフォルト: "D:\DailyBuild\Springhead2\test\report"
-h	Usage の表示
-v	バーバスモード (デフォルト)
-V	詳細バーバスモード
-s	サイレントモード
-D n	デバッグレベル
xxxx:yyyy	svn レビジョン番号の組 xxxx: 比較対象レビジョン (ファイル "revision.old" の内容が優先) yyyy: レポート対象レビジョン (ファイル "revision.new" の内容が優先)

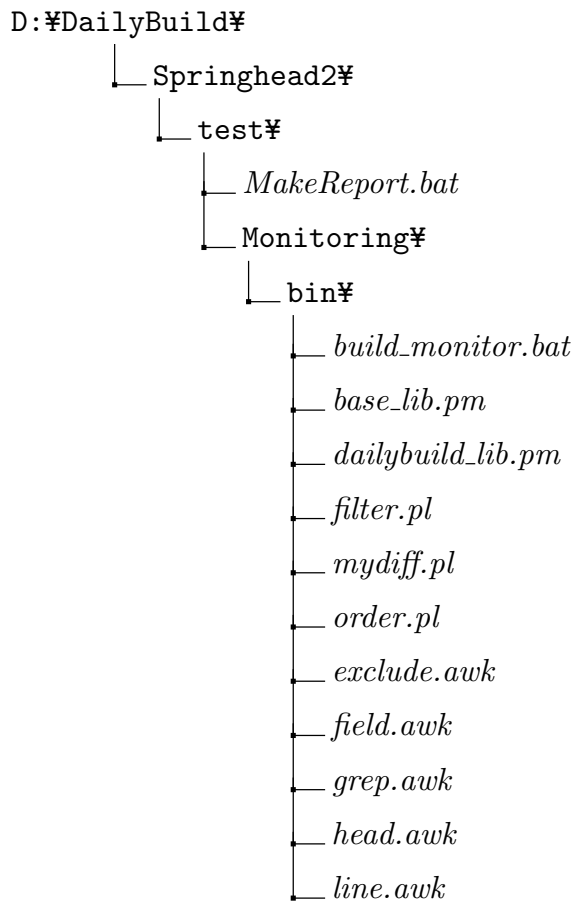
関連ファイル:

```
D:¥DailyBuild¥
├─ Springhead2¥
│   └─ test¥
│       └─ Monitoring¥
│           └─ etc¥
│               └─ revision.old
│                   └─ revision.new
```

これらのファイルには svn のレビジョン番号を記述する。

記述形式は "HEAD" または "r[0-9]+" (ファイルのコメントを参照)

関連スクリプト:



関連プログラム：



処理の説明：

1. ローカルホスト上でコマンドを実行

ディレクトリ "Monitoring" に移動して "build_monitor.bat" を実行し、レポートファイルを作成する。

引数は、次のオプション引数のうち指定のあったもの。

xxxx:yyyy	svn レビジョン番号の組
-k	作業ファイルを残す
-v	バーバスモード (デフォルト)

-V 詳細バーバスモード
 -D n デバッグレベル

詳細は“6.2.1 build_monitor.bat”を参照のこと。

2. コピー元/先のディレクトリの定義

コピー先ディレクトリ

REPBASE ← D:\DailyBuild\Springhead2\test\report

WABBASE ← \\haselab\HomeDirs\WWW\docroots\springhead\daily_build

3. ウェブにコピーするファイルの定義

コピーするファイルのローカルでの名称とウェブ上での名称は次のとおり。

"[日付].report" → "[WEBBASE]¥Test.report"
 "[日付].bldlog.diff" → "[WEBBASE]¥report¥Build.log.diff"
 "[日付].stblog.diff" → "[WEBBASE]¥rerpot¥StubBuild.log.diff"
 "[日付].runlog.diff" → "[WEBBASE]¥rerpot¥Run.log.diff"
 "[日付].spllog.diff" → "[WEBBASE]¥rerpot¥SamplesBuild.log.diff"

4. ウェブにコピー

3 に示したとおり、ファイルをウェブサーバにコピーする。

5. ローカルのファイルを削除

オプション引数-k が指定されていない限り、ローカルに作成したファイルを削除する。

6.2.1 build_monitor.bat

引数で指定した 2 つのレビジョンについて、svn からビルド結果と実行結果を取り出してレポートを作成する。

オプション引数：

-k 作業ファイルを残す
 -h Usage の表示
 -v バーバスモード (デフォルト)
 -s サイレントモード
 -D n デバッグレベル
 xxxx:yyyy svn レビジョン番号の組
 xxxx: OLD レビジョン (ファイル "revision.old" の内容が優先)
 yyyy: NEW レビジョン (ファイル "revision.new" の内容が優先)

出力ファイル：

\DailyBuild\Springhead2\test\report\Test.report

\DailyBuild\Springhead2\test\report\YYYY-MMDD.stblog.diff

```
\DailyBuild\Springhead2\test\report\YYYY-MMDD.bldlog.diff
\DailyBuild\Springhead2\test\report\YYYY-MMDD.runlog.diff
\DailyBuild\Springhead2\test\report\YYYY-MMDD.spllog.diff
```

関連スクリプト：

“6.2 MakeReport.bat” と同じ

関連ファイル：

“6.2 MakeReport.bat” と同じ

関連プログラム：

“6.2 MakeReport.bat” と同じ

処理の説明：

1. 下位処理プログラムのオプションの定義

ビルド結果の差分をとるための下位プログラムへ渡す引数を設定する。

MYDIFFOPT mydiff.pl へ渡す引数 4. 参照

FILTEROPT filter.pl へ渡す引数 4. 参照

2. 現在の日付と時刻

DATESTR の形式は"YYYY-MMDD hh:mm:ss"

3. 結果を報告するメールに関して

今のところメール機能は使用していない。

メール機能は Windows では動かない (/usr/sbin/sendmail を使うから)

4. 使用するディレクトリの定義 (丸括弧内は設定変数名)

```
Springhead2¥
├── test¥
│   ├── Monitoring¥
│   │   ├── bin¥ ... 関連するプログラムとスクリプトを配置 (BINDIR)
│   │   ├── etc¥ ... 関連するデータファイルを配置 (ETCDIR)
│   │   └── report¥ ... 作成するレポートファイルを配置 (REPDIR)
│   └── tmp¥ ... 作業領域 (TMPDIR)
```

5. 使用するプログラムの定義 (丸括弧内は設定変数名、角括弧内は使用言語)

order.pl Visual Studio が出力したビルドログの内容を、モジュール毎、スレッド毎、出現順に並び替える。(ORDER)[perl]

ライブラリ base_lib.pm, dailybuild_lib.pm を使用する。

mydiff.pl order.pl で並べ替えたログファイル 2 つについて差分をとる。

	差分処理はモジュール単位で行なう。(MYDIFF)[perl]
filter.pl	mydiff.pl でとった差分情報について、diff -c 形式に倣った出力情報を作成する。(FILTER)[perl]
nlconv.exe	改行コードの変換 (使用しない)
nkf.exe	漢字コードの変換 (使用しない)
field.awk	sep で行を分解し、field 番目の欄を返す。[gawk]
grep.awk	pat=rev なら "r[0-9]+" で始まる行を、pat=head なら "HEAD" で始まる行を返す。[gawk]
line.awk	line 番目の行を返す。[gawk]
exclude.awk	レポートに不要な行を削除する (現在はしていない)。[gawk]

6. レビジョン範囲の決定

この時点で変数 ARGV が 1 ならば (それは xxxx:yyyy のはずなので)、xxxx と yyyy をそれぞれ OLDREV と NEWREV に取り出す。

7. 次の svn(URL) を使用する

SVNURL svn のトップ URL
STBLOGURL "StubBuild.log" の URL
(以下同様)

8. 使用するファイル名 (入力側)

OLDREVFILE 比較基準レビジョンを指定するファイル
NEWREVFILE 最新のレビジョンを指定するファイル
SVNLOGTMP 作業ファイル

9. シグナルトラップの設定

未使用

10. 比較の基となるレビジョン (old-revision) の決定

引数で OLDREV が指定されなかったなら %OLDREVFILE% から読み出す。

11. 比較するレビジョン (new-revision) の決定

引数で NEWREV が指定されなかったなら %NEWREVFILE% から読み出す。

12. svn から %OLDREV% の日付と時刻を取得する

OLDDATE と OLDTIME に設定する。

13. svn から %NEWREV% の日付と時刻を取得する

NEWDATE と NEWTIME に設定する。

14. 出力するファイル名を決める (%NEWDATE% を用いる)

レポートファイルは %REPDIR% に、作業ファイルは %TMPDIR% に作成する。

ファイル名 = {stb|bld|run|spl}[err]log.{old|new|diff}

stb ライブラのリビルド

bld	src/tests のビルド
run	src/tests の実行
spl	src/Samples のビルド
err	上記の中でのエラー情報
old	%OLDREV% 関連
new	%NEWREV% 関連
diff	%OLDREV% と %NEWREV% との差分関連

stb,bld,run,spl 以外のテストを組み込むときは適宜名前を追加すること。

15. %OLDREV% と %NEWREV% のログファイルを取り出す
svn からログ情報を取り出し、モジュール単位かつスレッド順に並べ替える。
16. 差分をとる
%OLDREV% と %NEWREV% との差分をとる。(作業ファイル)
17. 参考情報を作成する (ShiftJIS)
16. のうち、エラー情報以外を参考情報ファイルにコピーする。漢字コードの変換はしていない (Windows で実行するから不要)。
18. レポートファイルにまとめる
レポートファイルには次の項目を記載する。
 - (a) 作成日付
 - (b) 新旧レビジョン番号
 - (c) ライブラリのビルドエラーログ
 - (d) src/tests のビルドエラーログ
 - (e) src/tests の実行ログの新旧差分
 - (f) src/Samples のビルドエラーログ
19. 改行コードを DOS 形式にする (ShiftJis 漢字が含まれているので Windows で見る)
実行しない (コメントアウト)
20. 結果をメールする
実行しない (コメントアウト)
メール機能を活かすなら、ここのコメントを外すだけでなく、REPORTBYMAIL の設定を見直すこと (現在 REPORTBYMAIL は強制的に 0 に設定されている)。
21. 後始末をして終了
作業ファイルを削除して終了する。ただし -k オプションが指定されていたら、作業ファイルはそのまま残す。

6.2.2 base.lib.pm

perl スクリプトで共通に使用するユーティリティ関数のライブラリ (§ を付したものは外部非公開である)。

説明：

`assert()`

アサーションルーチン。条件が満たされなければスタックをトレースして終了する。

`verbose()`

`verbose_set()`

バーバス表示ルーチン。`verbose_set()` で on/off を切り替えできる。

`show_usage()`

`add_padding()` ‡

使用方法の表示ルーチン。`add_padding()` は表示位置を合わせるための簡易ルーチン。

`warning()`

`error()`

`fatal()`

`panic()`

エラーメッセージ表示ルーチン。`fatal()` と `panic()` はプログラムを終了させる。

`dirname()`

引数のパス名からディレクトリ部分を取り出す。

`leafname()`

引数のパス名からリーフ名部分を取り出す。

`debug()`

`debug_push()`

`debug_pop()`

デバッグ表示を行なう (現在のデバッグレベルより大きいレベルが指定された時のみ)。デバッグレベルを変更するには `debug_push()` を使用する。`debug_pop()` で一つ前のデバッグレベルに戻せる。

`object_dump()`

引数で指定したオブジェクトをダンプする。表示には `debug()` を使用するので、デバッグレベルの指定が可能 (というか必須)。

6.2.3 dailybuild_lib.pm

Daily Build の結果解析で共通に使用する関数のライブラリ

説明：

read_log()

outside_threads_before()

outside_threads_after()

指定されたログファイルを読み込み、モジュール毎にスレッド順に整理する。

1. 関数値は次の三つ組。

%modules モジュール名とスレッド情報を関連付けた連想配列。キーはモジュール名、値は %threads の参照。ここで %threads はスレッド情報を格納する連想配列で、キーはスレッド番号、値はそのスレッドに属するログ行。

@modules モジュール名を出現順に格納した配列。

@lines ログの各行を出現順に格納した配列

2. モジュールの開始パターンは 3 種類あり、どれを使うかは引数で指定する。
パターン 1 "Build.log" 及び "Run.log" 用
パターン 2 "StubBuild.log" 用
パターン 3 "filter.pl" 用
3. スレッド関連より前に出力されたログ行は、スレッド番号 -1 のスレッドに属するものとして扱う。
4. スレッド関連より後に出力されたログ行は、スレッド番号 999 のスレッドに属するものとして扱う。

associate_modules()

mod_ins()

mod_del()

引数で指定された 2 つのモジュール配列 (@module1, @module2 とする) の対応する要素が同じモジュールとなるように各配列の要素を調整する。

1. @module1 にしかないモジュールに対しては、@module2 に ‘_DEL_’ マークを挿入することで対応をとる。
2. @module2 にしかないモジュールに対しては、@module1 に ‘_INS_’ マークを挿入することで対応をとる。
3. mod_ins() と mod_del() は、‘_INS_’ マーク及び ‘_DEL_’ マークを返す。

gather_lines()

モジュール名とスレッド情報を関連付けた連想配列から、指定されたスレッドのログのみを抽出する。rad_log() で返された %modules を処理の対象とする。