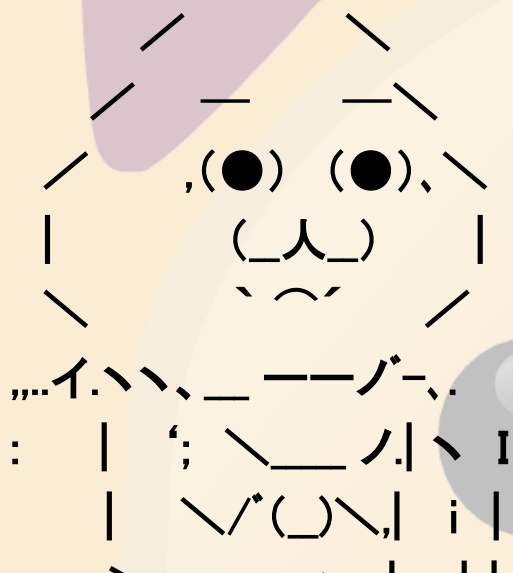
匠の伝承w

マルチな時代の設計と開発

パート7

スピーカー自己紹介



ゆーちです。

現役のエンジニアです。プログラム書いてます。

本名は、内山康広といいます。

31歳(H)です。おっさんです(笑)

株式会社シーソフト代表取締役です。

自宅警備員化してますが、ほっといてくださいw

BkReplyer を使いましょう(笑)。

http://www.vector.co.jp/soft/win95/net/se476325.html

Special thanks for 2ch.

本日の内容

- ログって見る。

  ログ全般の話
  ログ出力を実装する。

要求と技術的な反映方法を重点にお話します。
log4.net を使えばすむ、という話ではないです。

ログってどのように利用してますか？

- システムのログ

  OSのログイン／ログアウト、ネットワークアクセスの記録、

  ハードウェアやサービスの稼働状況
- アプリケーションのログ

  金融などで利用されるログ

  オペレータと金銭の動き
- デバッグ用のログ

  printf や TRACE の代用：恒久的に記録

大別するとこんなかんじ

1. セキュリティログ

ログイン、アクセスの認可、リソースの取り扱いなどに関する処理の記録

⇒セキュリティの診断、証拠の収集や解析

2. 業務ログ

OS、AP、主要業務の動作記録

⇒問題発生時の解析、障害時の復旧

3. ソフトウェアログ

・致命的なエラーや例外発生時の記録

・処理の続行に問題はないが、記録しておくべき警告レベル

・マウスやキーボードなどのオペレーションのトレース

・タスクやスレッドの切り替わり、状態遷移の記録

・通信などのメッセージ送受信の記録

・ある時点の変数の値や処理内容のダンプや記録

・ループなどの制御事象の記録

・単なるコメント

⇒デバッグを目的とするものが多い

ソフトウェアログ、だいじ。

3.（ソフトウェアログ）がしっかりしていれば、

1.（セキュリティログ）、2.（業務ログ）を吸収できるはず。

でも、世界中で共通の仕様があるわけではなさそう。会社やプロジェクトで

独自に実装されていて、その見方もそれぞれ固有の場合が多い。

ログ・・・こんなカンジになってませんか？

実際のログ出力（弊社ソースから抜粋）

[perl]

```
#配列生成の失敗時
$main::log->critical( "Fail to new CarrierContainer." ) if( $main::log );
```

[php]

```
/*内部変数のダンプ*/
$logger->variable('$result',$result);
$logger->info('$data:'.print_r($data,true));
$logger->info( "Read Config File .....OK" );
```

[C++]

```
//パケット送信内容の記録
String logString;
logString.sprintf( L"送信[%s][%d] [%s]", host_.c_str(), port_,
   message_.c_str() );
logTextQueue->Put( logString.c_str(), strlen( logString.c_str() ) );
```

ログ出力の要素

- __FILE__, __LINE__ ソースの位置、クラス名、メソッド名（デバッグ用）
- 記録された時刻
- コンテキストの位置
- ログインユーザ、オペレータ（認証記録、データ更新記録・調査用）
- 文字列。整形された文字列。（値のダンプもテキストとして出力）
- ログの区分（ログにはカテゴリレベルが必要）
  - 致命的なエラー（ディスクフル続行不能）
  - 一般エラー（超えた値の入力による続行不能）
  - 警告レベルのエラー（継続は可能だが問題があるかも知れない記録）
  - タスクのスイッチングや、制御の流れの記録（デバッグ用）
  - 通信電文や、要求内容とその応答内容の記録
  - 操作内容の記録（再現性試験で利用できるようなレベル、実行したSQL文とか）
  - デバッグ用に解説文やコメントを出力しておく
- カテゴリレベルは、動的に変化できると望ましい
  - 稼働中のアプリケーションにデバッグ分のログを出したり出さないようにする

ログをどこに出力するか？

- テキストファイル
- XML,CSV
- データベース
- LAN出力
- ランダムアクセスファイル（ローテーション）
  ⇒1つのログの長さを固定するか？

ログ出力の簡単なサンプル

- Example1

簡単すぎる(笑)

機能の追加

1. ファイル名、行番号を追加。
2. 日時を追加
3. コンテキスト情報
4. printf 形式で整形文字列を追加
5. レベルを追加
6. 動的なレベル変更機能（デバッグ時と稼働時でわけたくない）

このへんまで行ってみましょう。

＜パラメータの考察：クラスで吸収できるものとそうでないもの＞

＊const char　　*__FILE__ //フルパス名⇒△ログにはファイル名だけ
＊int　　line = __LINE__;
＊ELogLevel　　level = LOGLEVEL_ALWAYS;
△SYSTEMTIME　　sysTime; ::GetLocalTime(&sysTime);
△ContextId　　= ::GetCurrentThreadId();
△ContextInfo　　= (unsigned int)::GetCurrentThread();

Exapmle2

ここまでは、誰もが書きそうな内容です。

実際の業務で、このクラスが標準的なライブラリとして使えるでしょうか？

使えない理由

- 毎回 __FILE__, __LINE__ とタイプするの、めんどくせー。
- printf形式のフォーマット出力をサポートしたい
- 必要？不要？
  出力する内容はごもっともだがシステムには不要なデータもある
  出力形式は、こちらの都合に合わせたい。
- テキストファイル？バイナリファイル？
  intやDWORDの値をテキスト化して記録するなどリソースの無駄遣いだ。
  データベースに出力する、UDPでポート出力したい、など出力先変更
  出力方法は、プロジェクトの都合に合わせたい。

理想型を考えよう

・毎回 __FILE__, __LINE__ とタイプするの、めんどくせー。
・printf形式のフォーマット出力をサポートしたい

どのような記述形式だと、開発者の負担がないか。

＞記述を最小限にしたい。
LOG(L"ログ値(%d) ログメッセージ(%s)", value, text);
＞レベル設定は、次のように
TRACELOG(L"OnClick X(%d), Y(%d)", x, y);
DEBUGLOG(L"デバッグ=%s", text);

__FILE__, __LINE__ を書かない方法・・・
★マクロ定義で解決しようとすると・・・
#define LOG(msg) logger->Put(__FILE__, __LINE__, LOGLEVEL_ALWAYS, msg);
#define TRACELOG(msg) logger->Put(__FILE__, __LINE__, LOGLEVEL_TRACE, msg);
#define DEBUGLOG(msg) logger->Put(__FILE__, __LINE__, LOGLEVEL_DEBUG, msg);
msg に、printf 形式の可変引数をとれなくなってしまう・・・

#define LOG(format, ...) とは、書けない・・・

Exapmle3

P.S.

最近のC++拡張でマクロ定義に可変引数が渡せるようになりましたが、それを使っても実現できませんでした。

では、どうやるか。

Example3 で実装してみましょう。

Exapmle3のポイント

SLogWriter オブジェクトの生成過程、operator() でのパラメータ展開がミソ。

```
#define LOG       SLogWriter( Logger, __FILE__, __LINE__,
    LOGLEVEL_ALWAYS )
  :
#define DEBUGLOG     SLogWriter( Logger, __FILE__, __LINE__,
    LOGLEVEL_DEBUG )
  :
```

出力形式の変更

このままでは、まだ使えない。

テキストファイルの場合、

1：ファイルをずっとオープンしたままで良いの？

2：改行コードって 0d0a とは限らないんだけど。

3：マルチスレッドで動作しないじゃんw

4：うちは、ログはデータベースに書いてるから、このクラスは使えない

では、Example4 で、出力方法を切り離してみましょう。

Example4 のポイント

Adapter パターン

Loggerに必要だった、初期手続き、終了手続き、書き込み手順を仮想関数化

フォーマット変更を独自拡張したクラスで実装。

マルチスレッドへの対応もOKになりました。

データベースを利用する場合のクラスをどのように実装するかは、わかりますね？

その他

利用しているライブラリのもう一つの拡張

・すべて同じログ（ファイル）に出力するのか？

　大規模システムなどでは、サブシステムごとにログを振り分けたい。

　クラスごとに出力するログを変更

　LogManager による、アプリケーションログの一元管理

　アプリケーションには LogManager をリンクしておき、システム要件にあわせて

　アダプタクラスを再利用していけば、いつでも使えます。

ログを見たい

ログは、デバッグや、運用中のシステム障害の解析に必要不可欠なものです。

横河デジタルコンピュータの TRQer プログラムの動作ログを解析し、高速に
グラフ作成してデバッグ効率を効果的にします。
私も開発に関わっています。
制御システム開発している人は、いますぐ発注してください。
（笑）

http://www.yokogawa-digital.com/TRQer/

まとめ

今回は、ログを出力するというシーンを元ネタにして
汎用的なクラスライブラリにするための思考と実装の過程を説明してみました。

ソース位置情報や時刻情報の持ち方、渡し方（可変長の引数）
できるだけ記述を簡略にするためのマクロの利用と Writer クラスの拡張。
ほかのシステムでも使えるように、出力方法を切り離して実装する
Adapter クラス

C＋＋でしかできないワザもありますが、基本的な姿勢は、
「同じようなコードを2度以上書かないようにする」
ということを常に意識することだと思います。

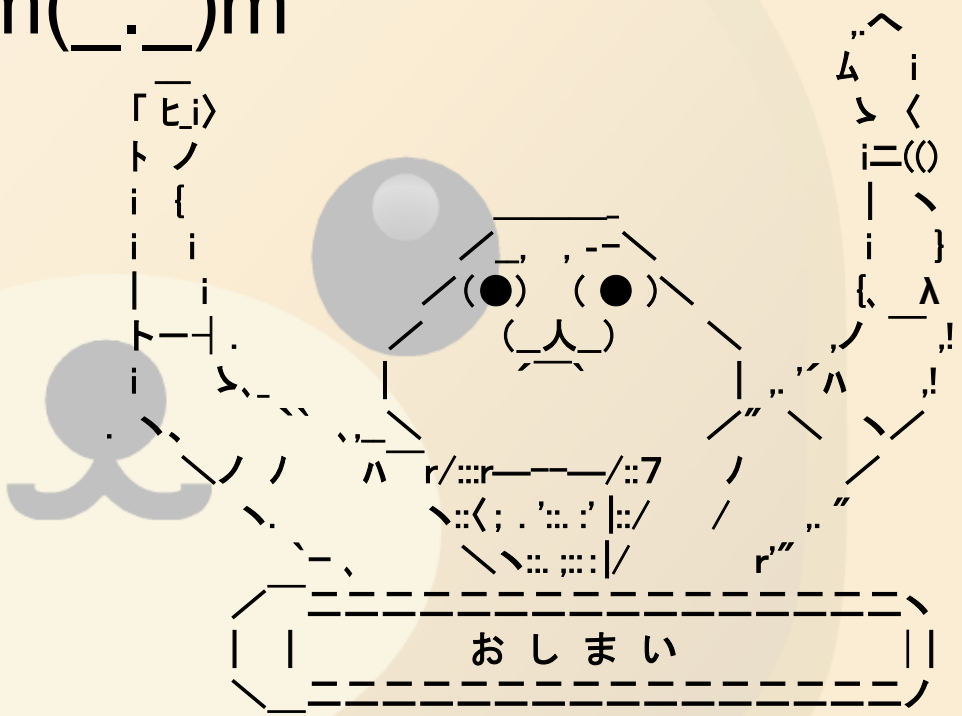
log4.net

DLLで、今開発したようなことが実装されてます（笑）

レイアウト定義も別に用意できるとか便利な部分も多いですね。
あちきは、いまんとこ、使わない・・・

ご静聴ありがとうございました。

m(_._)m



次回、実際の運用やログを見るための方法を解説するかも(笑)知れません。

Special thanks for Yaruo charactors