ＵＮＩＸ系システムプログラムのWindows/NTへの移植の一方式について

1Ｖ－5

仲田雅彦
（株）富士通神戸エンジニアリング　第三開発部

1. 今回の背景

当社製品のマルチプラットフォーム展開計画に基づき，ＵＮＩＸ系ＯＳ上で動作するプログラムをＭＩＣＲＯＳＯＦＴ®Ｗｉｎｄｏｗｓ／ＮＴ™に移植を行った．本稿では移植過程で発生した問題点と，今回解決した方式の一部を紹介する．

2. 移植における問題点

今回の移植で発生した問題は大きく以下の点にまとめられる．

－　ＣＰＵの違い
－　マルチプロセスとマルチプロセス／マルチスレッドの方式の違い
－　システム資源の違い

2.1　ＣＰＵの違い

Ｗｉｎｄｏｗｓ／ＮＴはＩｎｔｅｌ　Ｘ８６，ＤＥＣαに提供されている．ＵＮＩＸ系ＯＳでは当然サポートしていないＣＰＵであり，バイナリレベルでの互換は当然ない．また，エンディアンが違うため，他システムから転送されるデータの内容を取り出す場合，バイナリの構造をそのままあてはめた変数の形式で取り出すとソースレベルで互換がたもてなくなってしまう．

2.2　マルチプロセスとマルチプロセス／マルチスレッドの方式の違い

親と子の関係を保つのに，ＵＮＩＸ系ＯＳではマルチプロセスを，Ｗｉｎｄｏｗｓ／ＮＴではマルチスレッドを使用する，という違いがある．Ｗｉｎｄｏｗｓ／ＮＴでもマルチプロセスによる親／子関係の提供は可能だが，プロセスの生成にはスレッドの生成よりもさらに多くの資源が必要となる．また，Ｗｉｎｄｏｗｓ　ＳＯＣＫＥＴ等を使った場合に問題が発生することになる．

Ｗｉｎｄｏｗｓ　ＳＯＣＫＥＴはＤＬＬで提供されるが，使用時にスレッドを生成している．要するに他のプロセスへＳＯＣＫＥＴ環境を渡すことができなくなっている．

2.3　システム資源の違い

システムコールも当然違ったものとなっている．最たるものがプロセス間通信である，ＩＰＣとシグナルである．存在しないものは作成するしかなく，処理をうまく作成しておかないとプログラムの全面修正を必要とする．

An approach to the problem of porting UNIX system program to Microsoft Windows/NT 3.1

Masahiko Nakata
FUJITSU KOBE ENGINEERING LIMITED. 3rd software development dept.
6-9-1 MINATOJIMA NAKAMACHI CHUO-KU, KOBE-SHI, 650 Japan

3. 問題の解決方法

それぞれの問題を解決していくための一つのアプローチを以下の順に説明する.

－ エンディアンが違っても大丈夫なデータ構造

－ プロセスのリエントラントな構造

－ システムコールの本体処理からの分離

3.1 エンディアンが違っても大丈夫なデータ構造

基本的にバイナリデータは互換がないことを前提に処理を作成する. コンパイラのプリプロセッサ機能等を正確に使用してバイナリデータを本体処理に依存しない方式で処理する.

3.2 プロセスのリエントラントな構造

スレッドはリエントラントな構造になっている必要がある. プロセスを前提としたプログラムは複数同時に動作しても互いの環境を修正することがないため, リエントラントでない設計をされる可能性が大きい. 初めからリエントラントにしておけば, 当該部分をスレッドに置き換えても非常に少ない修正でマルチスレッドに対応することが可能となる.

3.3 システムコールの本体処理からの分離

明らかにＡＮＳＩ準拠でない関数－たとえＵＮＩＸ系ではあたりまえの関数だとしても－は本体処理で使用しない. 特に, ＵＮＩＸ系ではオブジェクトディスクリプタを共通に扱えてしまうが, これを明確にわけるようにしておくことにより, より一層移植性を高めることが可能となる.

4. おわりに

基本的にマルチプラットフォームを前提にしたシステムプログラムを開発しても, 論理上, システムの資源に頼るしかない場合もある. この時に, 当該システムだけをターゲットとした開発をしてしまうと, この時点でマルチプラットフォームの考えは崩れるわけである.

開発者はターゲットに重点をおいてしまうことが常である. これを制御するのがプロジェクトリーダであり, 方向づけをいかに正確に行えるか, によってシステムプログラム作成をよりスムーズに, より移植性の高いものを作成することができる. すなわちこれは, プロジェクトの成功へつながるわけである.

現在, 開発スピードを上回るスピートで新たなＯＳが発表されていく. これに対応するためには, プラットフォームを選ばないプログラム構造, 仕様である. 基本のプログラムを作成する時には, 各システムの要件をみたし, 許す限りの充分な時間をかけて設計することが重要であると考える.

〔参考文献〕

Inside Windows/NT, Microsoft