

4章： 問題の再現

ANDREAS ZELLER 著 中田秀基 監訳

デバッグの理論と実践

-なぜプログラムはうまく動かないのか-

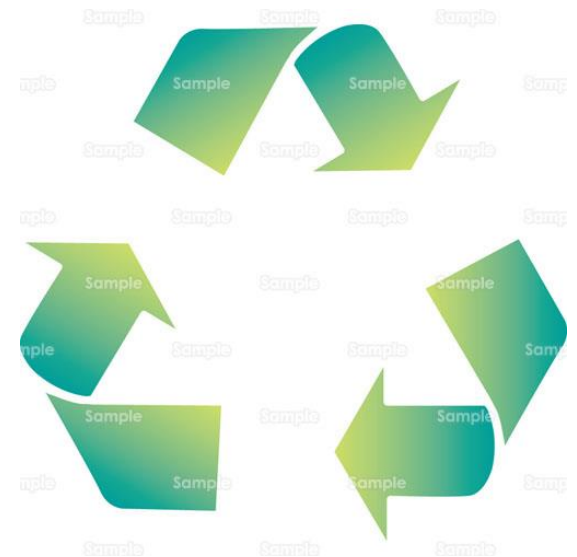
学籍番号 60206299

氏名 吉富 楓雅

目次

1. 問題環境の再現
2. プログラム実行の再現
3. 動作環境の再現
4. デバッグツールの影響
5. まとめ

問題環境の再現



デバッグの最初の作業

デバッグの第1歩： 問題の再現

- ・問題を観察するため
- ・問題が修正できるかどうかを確認するため



pixta.jp - 4620326

どうすればテストで特定の問題を再現できるか？

問題の再現

デバッグプロセスの中でも最も難しい問題の1つ:
特定の問題を再現するようなテストケースの作成

問題環境での作業は問題を再現するのに最も適しているが...

- ・プライバシー
- ・開発の容易さ
- ・メンテナンスのコスト
- ・移動のコスト
- ・実験のリスク

プログラム実行の再現

```
GType
thunar_application_get_type ()
{
    static GType type = G_TYPE_
    if (G_UNLIKELY (type == G_
    {
        static const GTypeInfo info =
        {
            sizeof (ThunarApplicationClass),
            NULL,
            (GClassInitFunc) thunar_applicati
            NULL,
            sizeof (ThunarApplication),
            0,
            (GInstanceInitFunc) thunar_applic
            NULL,
        };
        type = g_type_register_static (G_T`
    }
    return type;
}
```

SOURCE

実行に影響する入力

- 静的データ
- ユーザ入力
- コミュニケーション
- 時間
- 乱数
- 動作環境
- スケジューリング

意図が介在しない入力

- ・物理現象

宇宙線・放電・量子効果など

- ・デバッグツール

プログラムの実行に干渉するが問題を隠してしまうことがある

動作環境の再現



C++のパスワード要求プログラム

```
using namespace std;

string secret_password = "secret";

int main()
{
    string given_password;
    cout << "Please enter your password: ";
    cin >> given_password;
    if (given_password == secret_password)
        cout << "Access granted." << endl;
    else
        cout << "Access denied." << endl;
}
```

Password.cのstraceログ

※一部抜粋

write(1, "Please enter your password: ", 28) = 28

read(0, "secret\n", 1024) = 7

write(1, "Access granted.\n", 16) = 16

exit_group(0) = ?

デバッグツールの影響



再現可能と再現不可能

- ・**ボーアバグ**：

再現可能なバグ

- ・**ハイゼンバグ**：

バグを特定しようとするとうえてしまうバグ

- ・**マンデルバグ**：

根本原因が複雑かつ不明瞭で非決定的に見えるバグ

- ・**シュレーディンバグ**：

そのプログラムが動くはずがないということに気づくまでは出現しない

まとめ



まとめ

- ・問題のあった環境で再現を行うことは難しい.
- ・デバッグを行う上で「入力」の再現は重要である.
- ・バグには再現可能なものと不可能なものがある.