J4 情報システム実験実習 II　実験報告書

題目　　Web プログラミング

実施年月日　２０２５年１０月０３日

２０２５年１０月１７日

２０２５年１０月２９日

提出年月日　２０２５年１２月２０日

提出者

通し番号２１　学籍番号２２０５９　氏名　来間　空

1. 実験結果 1

bash の設定ファイルである~/.bashrc をカスタマイズした. .bashrc 内のカスタマイズした部分を図 1 に示す.

```
#設定 1（コマンドとしてディレクトリ名を実行すると，それが cd コマンドの引数として実行される）
shopt -s autocd

#設定 2（プロンプトの標準の表示からホスト名を削除する）
PS1='\u:\w\$ '

#設定 3（環境変数の LESS で--no-init 以外のオプションを指定する）
export LESS='-R -M -i -S -F'

#設定 4（Linux の教科書で例示されていない設定を 1 つ以上含める）
bind 'set completion-ignore-case on'          #設定 4-1（大文字小文字を区別しない）
bind 'set show-all-if-ambiguous on'           #設定 4-2（曖昧な場合は全候補を表示）
```

図 1　.bashrc 内のカスタマイズ

設定 1 は引数（ディレクトリ名）を入力するだけで，そのディレクトリに移動できる機能である．具体的には，shopt コマンドは bash のシェルオプションを表示・設定するコマンドであり，-s オプションは指定したシェルオプションを有効にするというものである．その引数 autocd は，ディレクトリ名のみで cd コマンドを実行するという機能である．

設定 2 はプロンプトの表示を決める設定であり，プロンプトの標準表示からホスト名を消去している．それぞれのシーケンスは，\u：ユーザ名，\w：カレントディレクトリを絶対パスで表示する，\$：権限（root なら「#」，それ以外なら「$」）を示すものである．これを実行すると，ターミナルでは「username:~/Desktop$」のように表示される．

設定 3 は export コマンドで指定したシェル変数を環境変数にするというものであり，これを実行すると，例えば「less ファイル名」と入力したとき，自動的に「less -R -M -i -S -F ファイル名」と実行されるのと同じ意味になる．R オプションはカラーコードを解釈して色付きのテキストを表示する機能である．M オプションは行数などの詳細なステータス情報を追加するものである．i オプションは検索時に大文字と小文字を区別しないというものである．S オプションを使用することで，行を折り返さずに表示できる．F オプションはスクロールの必要がない 1 画面に収まるファイルをページャー経由で表示せず，ターミナル内でそのまま表示する機能である．

設定 4 は独自にカスタマイズしたものであり，設定 4-1，4-2 はそれぞれ bash のタブ補完に関する設定である．設定 4-1 は引数の大文字と小文字を区別しないというものであり，設定 4-2 は補完候補が複数存在する場合，デフォルトではタブを 2 回押さないと表示されないが，タブ 1 回のみで表示されるというものである．

これらのカスタマイズを追加しようと思った理由は，前述した cd コマンドや less コマンドのカスタマイズによるものである．これらのコマンドは以前から使用機会が高く，さらにカスタマイズすることで効率化できたため，使用機会がより増えると感じた．したがって，それらに起因する引数の補完についてもカスタマイズを行うことで，さらにコマンドの効率を上げることができると感じたため，このカスタマイズを実装した．

2. 実験結果 2

作成した HTML ファイルはディレクトリ内の 21-22059-js.html であるため，それを参照すること．JavaScript のコードを交えて作成した html ファイルは今日の日付，現在の時刻を表示し，ストップウォッチ機能と任意のキーを押している間の時刻表示の機能を備える．ファイルの構成として，核となる body タグは，p タグや button タグによる画面表示部分と，その画面表示に対する機能を実装する script タグに分けられる．script タグは JavaScript で記述されており，このファイルの中で最も重要な部分である．

script タグの中は今日の日付と現在時刻の表示をする機能，ストップウォッチ機能，任意のキーを押している間は現在時刻を表示する機能の 3 つがそれぞれ独立して記述されている．

今日の日付，現在時刻の表示をする機能は，まず，日時のオブジェクトを生成する．今日の日付を表示するには年，月，日を取得しそれぞれ変数に格納し画面に表示する．現在時刻も同様に，時，分，秒を取得し，画面に表示する．時刻はリアルタイムで更新するようになっている．

ストップウォッチ機能はボタンが押された時に実行される．ボタンが押されたらミリ秒単位で計算する．これはボタンが押された時の時刻とストップウォッチを開始してからもう一度ボタンを押した時の時刻の差を求めることで実現している．また，ストップウォッチの経過時間が 4.900 秒から 5.100 秒の間であった場合「！ほぼ 5 秒！すごい！」と画面に表示される機能も備える．この機能が実際に表示されている画像を図 2 に示す．

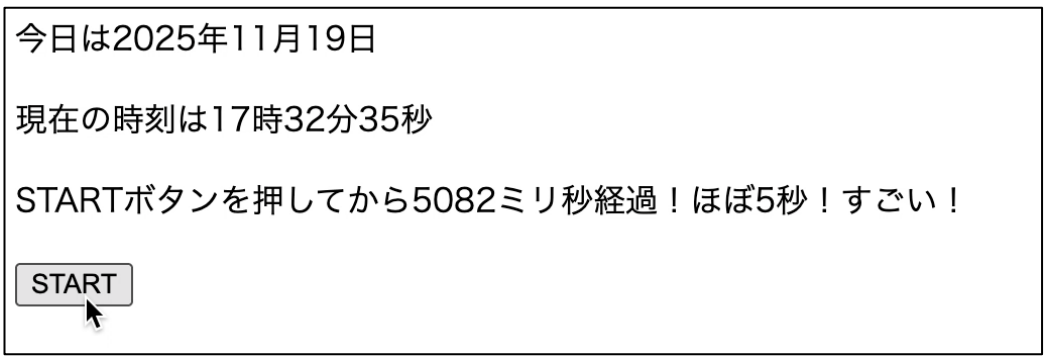

図 2　ストップウォッチ機能を実行した時の画面

押下による時刻表示機能は，任意のキーを押している間は時，分，秒を二桁表示でコロン区切りにして表示する．キーが離されたときは，「00:00:00」のように表示する．

3. 実験結果 3

作成した PHP ファイルはディレクトリ内の 21-22059-php.php であるため，それを参照すること．PHP ファイルは，アクセスしたときにまずサーバ側で今日の日付と現在の時刻を表示し，その下に ID とパスワードを入力するフォームを表示する．フォームには登録，認証，削除の 3 つのボタンがあり，それぞれが押されたときにサーバ側でデータベースと連携した処理を行う．

登録ボタンが押された場合は，ID またはパスワードが未入力なら登録せずにメッセージを表示し，両方入力されていれば MySQL データベースの web 表に同じ ID が存在しないかを確認したうえで，新規レコードとして ID とパスワードを登録する．

認証ボタンが押された場合は，入力された ID とパスワードの組み合わせが web 表に存在するかを調べ，一致すれば認証成功，そうでなければ認証失敗と表示する．

削除ボタンが押された場合は，同じく ID とパスワードの組み合わせが存在するかを確認し，存在する場合のみ該当レコードを削除し，結果をメッセージとして表示する．

4.　課題 1

bash，zsh はいずれも Unix 系システムで使われるシェルであるが，それぞれの特徴と用途が異なる．bash（Bourne Again Shell）は sh（Bourne Shell）の改良版として開発された，最も広く使われているシェルの一つであり，多くの Linux ディストリビューションで標準のシェルとして採用されている[1]．対して，zsh は bash の機能を取り込みつつ，さらなる拡張機能を追加した強力なシェルである．近年では macOS のデフォルトシェルにも採用されており，事例[2]に示されるように，zsh は macOS Catalina（2019 年）よりデフォルトシェルとなっている．

bash は基本的なタブ補完機能を持つのに対し，zsh はより高度で柔軟な補完システムを持つ．例えば，bash において，cd コマンドで補完候補が複数ある場合はタブを 2 回押す必要があるが，zsh では Tab 1 回で一覧と選択 UI が表示され，候補が種類ごとにグループ化されるため目的のディレクトリを素早く選ぶことができる[3]．また，コマンド履歴を確認したいときに，bash では直前のコマンドならカーソル上キーを用いるなどの基本的なコマンド履歴機能があるが，zsh はそれに加え，履歴の各コマンドに実行日時を紐づけて記録するタイムスタンプ機能や複数のターミナルやウィンドウ間で履歴を共有する共有ヒストリ機能を備える[3]．

このように，bash は安定性と高い互換性を重視しているのに対し，zsh は高い拡張性とカスタマイズ性があることがわかる．つまり，初心者や特にこだわりのない人は bash，コマンドラインの効率化やカスタマイズをしたい人は zsh を使用するのが適切である．

以上の比較結果を踏まえ，私の利用目的に最も適合するシェルは zsh ではないかと感じた．

最も大きな理由は私が主に使用しているパソコンが macOS であり，すでに zsh を利用しているからである．前述したように，zsh は macOS のデフォルトシェルとなっており，私のパソコンのシェルも zsh である．今後も macOS のパソコンを使用していくという保証があるため，zsh を使用するのが適切だと感じた．しかし，この実験で学習するまでは，そもそもシェルの種類以前にシェル自体について意識したことがなかった．もちろん，zsh はさまざまなプラグインやフレームワークがあるが，それらについても触れたことがなかった．このような経験を踏まえると，使用環境が acOS ということを除けば，互換性が高く初心者向けの bash を使うべきだと最初は考えた．しかし，今回の実験を通して，zsh の特性と bash との違いを知り，zsh を使用すべきだと改めて感じた．実際に，「Oh My Zsh」と呼ばれる zsh のフレームワークを導入し，それを用いてプラグインの「zsh-autosuggestions」やプロンプトテーマの「powerlevel10k」をインストールした[4][5][6]．インストール後のターミナルの実行画面を図 3 に示す．

図 3　zsh プラグインのインストール後のターミナル実行画面

図 3 より，zsh のプロンプトテーマである「powerlevel10k」を導入した結果，パス表示やプロンプトに装飾が追加され，情報の視認性が向上していることがわかる．また，末尾の行のカーソル右側に灰色の薄字で表示されている部分がプラグインの「zsh-autosuggestions」のインライン提案である．これにより，過去のコマンド履歴に基づく自動補完機能を実現することができた．このように，ターミナルの拡張性と macOS との互換性の観点から，bash よりも zsh を使用すべきだと考える．

5. 課題 2

Web プログラミングとは，Web サイトや Web アプリケーションを開発するために行うプログラミングのことである[7]．主に，ユーザが直接見て操作するフロントエンド開発と，ユーザから見えない部分の処理を実行するバックエンド開発に分けられる．例えば，昨年度の情報システム実験実習 I では，J コースの Web ページを作成する課題に取り組み，HTML，CSS，JavaScript といったフロントエンドの基本技術について学習した．一方，今年度の実験では，バックエンドに関わる要素を学習した．

Web の仕組みとして，インターネットに接続されたコンピュータは，ユーザが使用する端末であるクライアントと，Web ページやアプリケーションを格納し提供するサーバに分けられる[8]．Web サイトを閲覧する際には，クライアントがサーバに対してリクエストを送り，サーバがそれを処理してデータを返すことでページが表示される．前述のように，バックエンド開発はサーバ側の処理を実行するものであり，これらの仕組みがどのように動作するかをプログラミングする役割を持つ．バックエンドでは，Ruby，Python，JavaScript，PHP，Java などの言語が用いられる[9]．また，2 年前に WordPress で個人の Web サイトを作成した際には，Xserver のレンタルサーバを利用して公開し，独自ドメインの取得を試みた経験がある．このように，CMS を利用すればノーコードでフロントエンドを構築でき，さらにレンタルサーバを利用することで，セキュリティ面を含めた運用環境を簡単に整えられると感じた．

Web プログラミングでは，フロントエンドとバックエンドが連携して処理を行うが，より高度な機能を実現するためには外部サービスとデータをやり取りする場面も多い．このような異なるシステム間の通信を円滑に行う仕組みが Web API である．例えば，地図の Web API を利用すれば，アプリケーション上で地図表示や位置情報検索が可能となる．また，翻訳の Web API を活用することで，アプリ上に翻訳機能を組み込むことができる．天気予報，交通情報，ニュースなど，さまざまな分野で Web API が利用されている[10]．

さらに，Web アプリケーションの開発を効率化するためには，Web フレームワークの活用が基本となる．Web フレームワークとは，Web アプリケーションの作成，保守，拡張を容易にするための仕組みを提供するソフトウェアフレームワークである．Ruby の「Ruby on Rails」，Python の「Django」，PHP の「Laravel」などが代表的な例である[9]．これらのフレームワークを導入することで，ページ表示やフォーム処理，URL のルーティング，データベースとの連携，セキュリティ対策などが標準機能として提供され，開発者は基本的な処理を 1 から実装する必要がなくなる[11]．これにより，開発効率が向上し，コードの保守や機能追加も容易になる．

Web プログラミングにはデータベースの構築も欠かせない．データベースとは，情報を検索しやすく整理して保存するための仕組みであり，Web アプリケーションではユーザ情報，商品データ，投稿内容などの継続的なデータを安全に管理するために利用される．データベースは，MySQL，SQLite，PostgreSQL，Oracle Database，MongoDB などの RDBMS によって操作される[12]．

Web プログラミングにおいては，セキュリティに関する知識も重要である．Web サイトの脅威にはさまざまな種類があり，例えば，他のユーザのブラウザでスクリプトを実行させるクロスサイトスクリプティング（XSS）や，悪意のあるユーザがデータベースで不正な SQL コードを実行する SQL インジェクションなどが挙げられる[13]．これらの攻撃を防ぐためには，入力値の検証，エスケープ処理，認証や通信の安全性確保など，多面的な対策が求められる．適切なセキュリティ対策を実施することで，Web アプリケーションを安全に運用できる．

6. 課題３

Web アプリケーションセキュリティとは，Web サイトやアプリケーション，API を外部からの攻撃や不正利用から守るための仕組みや技術，運用の総称であり，最終的な目的は利用者の情報とサービスの信頼性を維持することである[14]. 現代の Web アプリケーションはログイン機能やデータベース連携など，多くの処理をブラウザとサーバの間でやり取りしているため，どこか 1 箇所に脆弱性があるだけで，サービス全体が危険にさらされる可能性がある．

Web アプリケーションにおける代表的な脆弱性の 1 つがクロスサイトスクリプティング（XSS）である．XSS は，攻撃者が悪意のある JavaScript などのスクリプトを Web ページに挿入し，そのコードがユーザのブラウザ上で実行されてしまう問題である．これにより Cookie やセッショントークン，入力中の情報など，ユーザが意図していないデータが外部に漏えいしたり，ユーザの操作が乗っ取られたりする．

私自身，ニュースサイトを閲覧しているときに突然「当選しました」「ウイルスが検出されました」と表示され，別サイトへ飛ばされた経験があった．また，スマートフォンで調べものをしていた際に，広告の閉じるボタンを押した瞬間にまったく別の怪しいページに移動したこともあり，ユーザ側の操作が悪意あるスクリプトで書き換えられていることを実感した．当時は単なる広告だと思っていたが，XSS の仕組みを学ぶことで，これらが悪意あるスクリプトによる誘導である可能性が高いと理解した．

XSS には反射型と保存型があり，反射型 XSS では，攻撃者が不正なスクリプトを含んだ URL をユーザに踏ませることで発生する[15]．この攻撃は，メール，SNS，掲示板などユーザが気軽にリンクをクリックしやすい環境で特に危険である．URL の後半に意味不明な文字列や長いパラメータが続いているのは典型的な特徴であり，私も実際に見かけて疑問に思ったことがある．

一方で保存型 XSS は，攻撃者がサイト内のコメント欄やプロフィールなどに不正スクリプトを埋め込み，それがデータベースに保存され，他のユーザがページを開いたときに自動で実行されるというものだ．この攻撃は長期間残り続け，閲覧したユーザ全員が被害を受ける可能性があるため，影響範囲が広く，さらに深刻である[15].

これらに対して開発者側が取るべき対策として，入力値検証（バリデーション），出力時のエスケープ処理，危険なタグや文字列の禁止，Content Security Policy（CSP）の導入などが挙げられる．例えば，script タグやイベントハンドラ（onclick など）が入力に含まれていないかを確認し，サーバ側で安全な文字列に変換して表示するだけでも被害を大幅に減らせる．さらに，CSP を導入することで，ページ内で実行できるスクリプトの範囲を制限でき，不正な JavaScript の実行を根本的に阻止できる[16].

ユーザ側でも，不審なリンクを踏まないこと，URL の構造を確認すること，信頼できない掲示板や SNS で表示される広告を警戒することなど，できる対策は多い．私も XSS について学んだ後は，URL の末尾を注意深く見るようになり，怪しいパラメータが続くリンクは開かないようにしている[16]．ブラウザを使用するときは不審なリンクを警戒するだけでなく，不審な広告を自動的にブロックするできる拡張機能をインストールし，使用している．

このように，Web アプリケーションの脆弱性には XSS のようなクライアント側を狙うものから，SQL インジェクション，CSRF，不適切なアクセス制御など，多様な種類が存在する．それぞれに対策が必要であり，セキュリティは「一度対策すれば終わり」というものではなく，開発者と利用者が継続的に取り組むべき課題である．Web アプリケーションにおける情報セキュリティとは，これらの対策や運用を通して，サービスの安全性と信頼性を守る取り組みそのものである．

7. 感想

今回の実験を通して，シェルの仕組みと Web の全体像を理解することができた．bash の学習では，設定ファイルである .bashrc をカスタマイズするシェルスクリプトを作成したことで，bash への理解が深まった．また，レポートの課題 1 において zsh と bash を比較するため，実際に zsh をカスタマイズした．これにより zsh への理解と，シェル全体に対する理解が深まり，良い経験になったと感じた．

また，昨年の実験で実施した「Web ページの作成」において，実際に Web ページを作成した経験があった．今回の実験で Web プログラミングを行ったことで，Web サイトの仕組みをより深く理解することができた．今までは利用者として Web サイトを使う立場だったが，今回は作る側の体験を通して，Web サイトの構成が複雑で，さまざまな機能が複合的に作用して動作していることを知り，非常に驚いた．また，Web サイトは作るだけでなく，セキュリティについても考慮する必要があることが分かり，セキュリティの重要性についても理解が深まった．

今回の実験を通して Web プログラミングにさらに興味を持ったが，まだ学習すべき点が多いと感じている．今後は，バックエンドやデータベースなどの分野についても学び，Web サイトを実際に公開するなど，実践的な経験を通して理解を深めていきたいと感じた．

参考文献

[1] DonTacos, 「bash と zsh と sh について」, note, https://note.com/mtng420/n/n4e9f6e8bc035, 2024 年 9 月 26 日公開.

[2] 「bash と zsh の違い。bash からの乗り換えで気をつけるべき 16 の事柄」, 節約テクノロジ, https://kanasys.com/tech/803, 2025 年 11 月 9 日参照.

[3] ねこねこ, 「【今更】bash と zsh の違いについて網羅的にメモ」, https://lazy-developer.jp/bash-zsh-differences/, 2025 年 3 月 13 日更新.

[4] 「Oh My Zsh」, https://ohmyz.sh/, 2025 年 11 月 11 日参照.

[5] 「zsh-autosuggestions」, GitHub, https://github.com/zsh-users/zsh-autosuggestions/blob/master/INSTALL.md, 2025 年 11 月 11 日参照.

[6] 「powerlevel10k」, GitHub, https://github.com/romkatv/powerlevel10k, 2025 年 11 月 11 日参照.

[7] 「Web プログラミングとは？必要な 7 つの知識や言語の種類、学習方法をご紹介！」, https://web-camp.io/magazine/archives/61012/, 2024 年 3 月 30 日更新.

[8] 「ウェブのしくみ」, MDN Web Docs, https://developer.mozilla.org/ja/docs/Learn_web_development/Getting_started/Web_standards/How_the_web_works, 2025 年 11 月 17 日参照.

[9] Sky 株式会社, 「フロントエンドとバックエンドの違いは？ 図や具体例を用いて徹底解説」, https://www.skygroup.jp/media/article/3309/, 2024 年 9 月 5 日公開.

[10] 「Web API とは？【初心者向け】基礎知識を徹底解説」, https://www.dal.co.jp/column/r-webapi/, 2024 年 8 月 1 日.

[11] 「サーバーサイドウェブフレームワーク」, MDN Web Docs, https://developer.mozilla.org/ja/docs/Learn_web_development/Extensions/Server-side/First_steps/Web_frameworks, 2025 年 11 月 17 日参照.

[12] 三宅 悠太, 「Web 開発に必要不可欠なデータベースや SQL とは？ SQL 習得に必要な基本のキを学ぶ」, Code Zine, https://codezine.jp/article/detail/16513, 2023 年 1 月 20 日更新.

[13] 「ウェブサイトのセキュリティ」, MDN Web Docs, https://developer.mozilla.org/ja/docs/Learn_web_development/Extensions/Server-

side/First_steps/Website_security#sql_%E3%82%A4%E3%83%B3%E3%82%B8%E3%82%A7%E3%82%AF%E3%82%B7%E3%83%A7%E3%83%B3, 2025 年 11 月 18 日参照.
[14] 「Web アプリケーションセキュリティとは？」, Cloud Flare, https://www.cloudflare.com/ja-jp/learning/security/what-is-web-application-security/, 2025 年 11 月 18 日参照.
[15] 「XSS 攻撃を防ぐ方法」, Cloud Flare, https://www.cloudflare.com/ja-jp/learning/security/how-to-prevent-xss-attacks/, 2025 年 11 月 18 日参照.
[16] 「XSS の仕組みと対策について」, エンベーダー, https://envader.plus/article/13, 2022 年 10 月 30 日.