Research Computing User's Guide Practice Key

November 07, 2022

Jason W. Bacon

Contents

List of Tables

November 07, 2022

0.1 Using this key

This key to the practice problems is provided to allow students to immediately *check their own work*. Do not look at this answer key before writing down your own answers. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams. Writing things in your own words vastly improves your understanding.

Chapter 1

Computational Science

1.1 Introduction

1.1.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What kind of research are you currently conducting, and how might computers be used to further your goals?

   No wrong answer.

2. How does computational science differ from computer science?

   Computer science is the study of computers and computing methods while computational science is the application of computers to other sciences.

3. What areas of research benefit from computational science?

   Virtually all of them.

4. Describe two reasons that computational science is growing so rapidly.

   - New technologies are becoming integrated into most areas of research and computers are needed to analyze the data they collect.
   - Information storage has become vast and cheap, making it possible to collect massive amounts of data about any subject.

5. What are the major goals in the computational time line?

   Minimize wall time, manual labor, and computing time.

6. What are the major steps in the computational time line? Which one takes the longest?

   Development time, deployment time, learning time, run time. Any one of them could be the longest step, depending on many factors.

7. Can researchers avoid programming entirely? Why or why not?

   Probably not. Commercial software and programmer time are too expensive and time consuming for most researchers.

1.2 Common Methods Used in Computational Science

1.2.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. Why are numerical analysis techniques so important to science and engineering?

   Most mathematical models of real world systems cannot be solved via known analytical methods.

2. Explain Newton's method for finding the roots of a function. Use a graph to illustrate.

   Start by guessing the value of x. Draw a tangent line to the graph of the function at x, f(x). Compute the root of this line (where it crosses the x axis) using the equation for a line with slope f'(x). This is the next guess. Repeat until successive guesses are very close to each other or until f(x) is close to 0.

3. What is computational modeling? Describe two examples of processes commonly modeled on computers.

   Using computers to model events happening in the real world. Models are used for fluid flow and traffic patterns among many other things.

4. What is data mining? What is one of the major challenges in designing useful data mining software?

   Searching through data for interesting patterns to be determined by the computer. The user does not tell the program exactly what to look for. This requires some level of intelligence in the software.

5. What is a parameter sweep? Describe one task that requires a parameter sweep. Is doing parameter sweeps desirable?

   Useful when the answer cannot be computed, but is easy to verify. Check every possible answer until one can be verified. A last resort for finding the answer to a question when numerical analysis or other methods cannot be used to home in efficiently.

6. What is data sifting? Describe one real-world example that requires data sifting.

   Searching through large amounts of data for a rare gem. The LIGO project searches massive amounts of laser interferometer data for evidence of gravity waves.

7. What is a Monte Carlo simulation?

   Using random data to represent a larger population.

8. Can you think of any computational science methods that do not fall into one of the categories described here?

   No wrong answer.

1.3 Venture Outside the Computer Science Bubble

1.4 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. How does the perspective of computer scientists and engineers often differ from that of scientific researchers? Which perspective is better?

   Researchers study problems, such as cancer, pollution, the origin of the universe, etc. and then look for or invent solutions to those problems using whatever resources they have available.

   Technologists study solutions (tools) such as specific programming languages, deep learning, GPU computing, computer-aided design (CAD), etc. and then look for problems that they can solve.

   Value judgments such as "better" are a foolish waste of time that only serves the ego. What we need to do, regardless of the perspective from which we begin, is match the best solution to each problem.

Chapter 2

Where do I get the Software?

2.1 But I Hate Programming...

2.1.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What are the three ways to obtain research software?

   Buy it, download it, or write it.

2. What kind of programming do most researchers need to learn? Why?

   Most researchers can get by running software written by others and some simple scripting of their own to automate pipelines, because there is a vast amount of software available for common analyses.

3. Why is it a good idea for researchers to learn how to program beyond simple scripting?

   There may not be any software to do the analysis you need, or existing software may be low-quality. Hiring a programmer is not feasible for most due to limited talent and high cost.

4. What is the benefit of learning a compiled language?

   It may run hundreds of times faster than scripting languages.

2.2 Buy It

2.2.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. List four disadvantages of commercial software vs free open source software (FOSS).

   - Expensive
   - Runs on only a few supported operating systems and CPUs
   - Requires license management
   - Restricted access to documentation and support prevents simple web searches

2. For whom is commercial software generally a good option?

   For those who can afford it and have needs that are not met by free open source software

2.3 Download It

2.3.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What are three advantages of FOSS (free open source software)?
   - It's free
   - No laborious license management
   - It runs on many operating systems and hardware platforms
   - Installation is trivial when using a package manager
2. What can the average computer user do with FOSS nowadays?

   Everything they need for typical computer use.
3. What is a caveman installation and when should one be performed?

   Manually downloading, patching, building, and installing software. It should only be done in the 1980s or earlier.
4. What is a package manager?

   A system for reliably installing, uninstalling, and upgrading a software package and everything on which it depends with one simple command.
5. What is an advantage of source-based package managers such as FreeBSD ports, Gentoo Portage, MacPorts, and pkgsrc, over binary-only package managers such as Debian packages and Conda?

   They allow custom builds with non-portable CPU features that might improve performance, as well as non-standard program features and options.

6. What are some of the problems that package managers solve when compared with caveman installations?

   See list in text above.

7. What can you do besides resort to caveman installations if your package manager doesn't have a package for your software?

   See list in text above.

2.4 Containers

2.4.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. Are containers "good"?

   As always, labeling things generally "good" or "bad" is a waste of time. Containers have many valid uses but are also misused as an alternative to writing quality software.

2.5 Finding Research Software

2.5.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. Is it a good idea to trust the advice of a colleague about what software to use? Why or why not?

   No. They may not be aware of better alternatives.

2. How can you be certain that you are using the best available software for your research?

   You must do a thorough investigation on the Internet AND discuss it with others.

2.6 Write It

2.6.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What are the primary goals in writing any software?
   - Leverage well-tested existing code. Write as little of your own code as possible so you don't reinvent the wheel.
   - Portability (runs on any hardware and operating system)
   - Performance (runs efficiently, especially for computational software)
   - Reliability (produces correct output, does not crash)
   - Maintainability (clean, simple code that's easy to read)
   - User-friendliness (easy to use, meaningful error messages)
2. What is the main advantage of compiled languages over interpreted languages?

   A compiled program will run roughly 100 times faster in many cases.
3. What is the main advantage of C over other compiled languages for busy researchers?

   C is a simple language that can be mastered without devoting a major portion of our life to it. Then we can focus on writing quality code instead of learning more language features.
4. How much should we rely on the advice of others when choosing a language or operating system? Why?

   Not much. Many people choose languages and operating systems for irrational reasons. We should investigate the features objectively and determine what will make our computational research as easy as possible.

2.7 Running Your Software

2.7.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

Pro: We can configure it any way we want and install any software we want.

Con: Time consumed by I.T. activities is taken away from our research.

1. What are the pros and cons of managing your own computer(s) for research vs using computers managed by your organization?

Chapter 3

Using Unix

3.1 KISS: Keep It Simple, Stupid

3.1.1 Practice

1. What's the engineer's motto regarding things that ain't broke?
   If it ain't broke, it doesn't have enough features yet.
2. Why do many people believe that Unix is hard to learn?
   Because some people like to make it sound hard in order to impress others (or themselves).
3. Is it better to accumulate vast amounts of knowledge or to become highly skilled using the fundamentals? Why?
   Fundamentals. Knowledge is not wisdom. Wisdom is knowing how to apply it effectively.
4. What is the core principle of Unix design? Explain.
   Simplicity. If is designed to take the simplest approach to each common task, even if it doesn't seem intuitive at first.

3.2 What is Unix?

3.2.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. After learning Unix, on what operating systems will you be able to use your new skills?

   Any operating system, including Windows with a compatibility layer such as Cygwin or WSL.

2. What is the major design goal of the Unix standards?

   Freedom of choice, so users can easily switch to a different Unix platform if the choose.

3. What is the alternative to learning Unix for computational scientists? Why?

   Relying on the charity of others. Most scientific software is written for Unix and most will never have a graphical user interface.

4. Why does most scientific software lack a convenient graphical or web interface?

   Economic reality. Such interfaces require many man-hours to build and maintain, and neither the resources nor the man-power exist.

5. Is Unix an operating system? Why or why not?

   No. Unix began as an operating system but evolved into a standard to which operating systems conform.

6. What is the advantage of open standards?

   They give consumers freedom of choice, to easily switch between competing products.

7. How many different Unix-compatible operating systems exist? What does this mean for Unix users?

   Too many to list. It means our investment in learning Unix and developing programs for Unix will be safe even if the Unix system we use now ceases to exist.

8. Which mainstream operating systems are Unix-compatible and which are not?

   Microsoft Windows is the only mainstream operating system that is not Unix-compatible.

9. What types of computer hardware run Unix?

   Everything from cell phones to PCs to high-end servers.

10. How much does Unix cost?

    Many high-quality Unix-compatible operating systems are completely free.

11. Which Unix operating system is the best one?

    This is a silly question. It obviously depends on our needs.

12. How should we go about choosing a Unix system? What if we make the wrong choice?

    By carefully assessing our needs and objectively examining the features of several alternatives to see which one matches the best. If we make the wrong choice, it will not be hard to switch later. Most of the Unix skills we learn and programs we use can be easily transferred to another Unix system.

13. How do we spot evangelists who are likely to give us irrational advice?

    They do not ask any questions about our needs and lack knowledge of multiple alternatives.

14. What is an API?

    API = Application Program Interface. It is a set of standards defining how programs interface with the operating system and other software.

15. What is the advantage of the Unix API over the APIs of non-Unix operating systems? What problem does it solve?

    The Unix API is common to many operating systems and hardware platforms, so programs developed for Unix will never have to be rewritten to run on different Unix systems or hardware.

16. Can software written for Unix be run on Windows? How?

    Yes, by using a compatibility layer such as Cygwin or WSL.

17. How does the Unix API help us proactively eliminate software bugs?

    It enables us to easily test new code on multiple Unix variants and hardware platforms, which exposes bugs that may not have been visible on the primary development platform.

18. What is a UI? What are three advantages of the Unix UI over the UIs of non-Unix operating systems?

    UI = User Interface. 1. The Unix UI is largely the same across all Unix platforms, so what we learn on on Unix system can be used on others. 2. The Unix UI is designed to be simple and elegant. 3. The Unix UI is highly consistent, so what we learn about one Unix command often applies to others.

19. Why are Unix-compatible operating systems faster, more stable, and more secure than many non-Unix platforms?

    They compete directly with each other, rather than trap customers with a proprietary API and UI that makes it difficult to switch to a new system. This means they have to focus on objective measures such as reliability, performance, etc.

20. How does the inherent remote access capabilities of Unix help researchers?

    It allows them to share computing resources, so there are fewer machines to maintain.

3.3 Unix User Interfaces

3.3.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is a UI?

   UI = User Interface. Software that allows users to interact with the computer.

2. What is a GUI?

   GUI = graphical user interface. Usually a mouse-driven or touchscreen interface with icons and menus.

3. What is the difference between Unix and other operating systems with respect to the GUI?

   Most Unix systems offer a wide variety of different GUIs from which to choose. Unlike Windows and macOS, we cannot talk about what Unix "looks like".

4. How is Unix + X11 different from remote desktop systems and terminal servers?

   It allows us to easily run graphical programs on multiple remote Unix machines, all displaying on the same desktop.

5. What is a virtual desktop?

   A separate desktop image, like having multiple monitors on a single physical monitor.

6. What are the two basic types of user interfaces? Which type is a GUI?

   Command-driven and menu-driven. A GUI is a menu-driven system where menu items may be either text or icons.

7. What is a CLI?

   CLI = Command Line Interface: A user interface that accepts commands typed in by the user and runs them immediately.

8. What types of applications are better suited for a menu-driven interface? Why?

   Those with limited functionality that can fit in one or a few menus, and those that are used infrequently since it would be hard to remember commands we don't use regularly.

9. What types of applications are better suited for a command-driven interface?

   Those that have too much functionality to fit in one or a few menus and those that we use every day, so remembering commands comes naturally.

10. Which is easier to automate, a menu-driven system or a CLI? Why?

    A CLI is easier to automate since a sequence of commands can simply be stored in a file (called a script).

11. How many scientific programs offer a menu-driven interface? Why?

    Very few. Menu-driven interfaces require a great deal of work to build and maintain, and the scientists writing the mostly free software don't have the time or skills for this.

12. What is a shell?

    A program that implements a CLI.

13. What is a kernel?

    The innermost layer of an operating system, which controls the hardware.

14. What are libraries? What kinds of functionality do they provide?

    Collections of subprograms that may be useful to more than one application, such as basic input/output, basic math, etc.

15. What is a terminal?

    A screen and keyboard used to interact with a computer.

16. What is a terminal emulator?

    A graphical program that pretends to be a hardware terminal.

17. Do people still use hardware terminals today? Explain.

    Not so much. They have mostly been replaced with terminal emulators running on PC desktops.

18. What is a shell prompt?

    The prompt is output from the shell indicating that it is ready for more input, i.e. the next command.

3.4 Still Need Windows? Don't Panic!

3.4.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. Describe three ways we can use Unix software on a Windows machine.

   (a) Remotely log into a Unix system using a terminal emulator such as PuTTY.
   (b) Run Unix in a virtual machine under Windows.
   (c) Use a Unix-compatibility layer such as Cygwin to run Unix programs directly under Windows.

2. What is the advantage of Cygwin over a virtual machine?

   It installs in about 10 minutes on a typical machine.

3. What is the risk of using Cygwin?

   None. It is completely isolated from the rest of the Windows system, so installing it will not cause any damage.

4. What are two advantages of a virtual machine over Cygwin? Explain.

   - Performance. Cygwin suffers from bottlenecks in system calls.
   - Full functionality. Not all Unix software will run under Cygwin.

5. What is an advantage of Cygwin over WSL?

   It is completely open source, so Cygwin users are not at the mercy of Microsoft for continued support.

3.5 Logging In Remotely

3.5.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What must be added to Unix to allow remote access?

   Nothing. This capability has been part of Unix since the beginning.

2. Can we run graphical programs on remote Unix systems? Elaborate.

   Yes, but some of them may require a very fast connection in order to function well.

3. Does the CLI require a fast connection for remote operation?

   No, it will work well even over very slow connections.

4. What command would you use to log into a remote system with host name "myserver.mydomain.edu" using the user name "joe", assuming you want to run a graphical X11 application?

   ssh -X joe@myserver.mydomain.edu

   -X may be replaced with -Y.

5. What should you do if someone advises you to use rsh or telnet?

   Stop trusting the person who gave this advice.

6. How can Windows users add an ssh command like the one used on Unix systems?

   Install Cygwin (or WSL).

7. What is the purpose of the TERM environment variable? What will happen if it is not set correctly?

   It informs the Unix system what magic sequence to send to your terminal and expect from your keyboard.

3.6 Unix Command Basics

3.6.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What are the three major components of a Unix command?

   Command name, flags, and data arguments.
2. What are the two sources of the command name?

   It is either the filename of a program or a command built into the shell.
3. How do we know whether an argument is a flag or data?

   In almost all cases, flags begin with a '-'.
4. What is the advantage of short flags and the advantage of long flags?

   Short flags are less typing, long flags are more easily understood.
5. What do data argument represent?

   Either the data itself, or the file or directory name containing the data.
6. What rules does Unix enforce regarding the order of arguments?

   None. This is left to the programmer to decide.
7. What separates one Unix argument from the next?

   White space, i.e. space or tab characters.
8. Can an argument contain whitespace? If so, how?

   Yes, the argument must be enclosed in quotes, or the whitespace must be preceded by a '\'.

3.7 Basic Shell Tools

3.7.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is the de facto standard shell on Unix systems?

   Bourne shell.

2. How do most Unix commands differ when run under one shell such as C shell as opposed to running under another such as Bourne shell or Bourne again shell?

   Not at all. Most commands are the same regardless of which shell we run them under.

3. What if a shell we like or need is not present on our Unix installation?

   It can probably be installed via the systems package manager in a few seconds.

4. How can we quickly rerun the previous command in most Unix shells?

   Just press the up arrow to scroll back through the command history and press enter after the command appears.

5. Show a Unix command that lists all recent commands executed from this shell.

   Run the **history** command.

6. Show a Unix command that runs the last command that began with "ls".

   !ls

7. Given the shell history shown below, show a Unix command that runs the last command used to log into unixdev1.

```
984  16:48   vi .ssh/known_hosts
985  16:50   ssh -X bacon@unixdev1.ceas.uwm.edu
986  16:52   ssh -X -C bacon@unixdev1.ceas.uwm.edu
```

```
  987  16:58   ape
  988  16:59   ssh -X -C bacon@unixdev1.ceas.uwm.edu
```

!988 or !ssh

8. How can we avoid typing a long file name or command name in most shells?
   Type part of it and press TAB for auto-completion.
9. How can we instantly move to the beginning of the command we are currently typing?
   Ctrl+a is common, but it depends on the shell and the user's customizations.
10. How do we list all the non-hidden files in the current directory ending in ".txt"?
    ls *.txt
11. How do we list all the hidden files in the current directory ending in ".txt"?
    ls .*.txt
12. How do we list all the files in /etc beginning with "hosts"?
    ls /etc/hosts*
13. How do we list all the files in the current directory starting with a lower case letter and ending in ".txt"?
    ls [a-z]*.txt
14. How do we list all the files in the current directory starting with any letter and ending in ".txt"?
    ls [A-Za-z]*.txt
15. How do we list all the non-hidden files in the current directory ending with ".pdf" or ".txt"?
    ls *.{pdf,txt}
16. How do we list all the files in /etc and all other directories under /etc with names ending in ".conf"?
    find /etc -name '*.conf'
17. When are globbing patterns normally expanded to a list of files?
    Before the command is executed.
18. How can we include a file name in a command if the file name contains a special character such as '*' or '['?
    Enclose the file name in quotes or escape the special character (precede it with a '\').

3.8 Processes

3.8.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. How does a process differ from a program?

   A program is a file with statements or commands in it. A process is the execution of a program. There can be many processes running the same program.

2. If 10 people are logged in and using the same Unix shell, how many shell programs are there? How many shell processes?

   1 program, 10 processes.

3. What normally happens when you run a program from the shell?

   The shell creates a child process to run the program and waits for that process to terminate before printing the next shell prompt.

4. How are processes identified in Unix?

   Each has a unique integer process ID.

5. Show a Unix command that lists all the processes currently running on the system.

   ps -ax

6. Show a Unix command that monitors which processes are using the most CPU and memory resources.

   top

3.9 The Unix File System

3.9.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is a file in the viewpoint of Unix?

   A sequence of bytes, nothing more.
2. What is the difference between a text file and a binary file?

   Text files contain only ASCII/ISO characters while binary files may contain both characters and non-character data.
3. What will happen if you echo a binary file to your terminal?

   The terminal will try to interpret the bytes as ASCII/ISO characters and it will look like nonsense.
4. What is the difference between Windows and Unix text files?

   Unix uses only a newline character to terminate each line while Windows uses both a newline and a carriage return.
5. How can we convert text files between the Unix and Windows standards?

   Using the free dos2unix and unix2dos commands.
6. What is a directory?

   A file system object that "contains" files and/or other directories, or more accurately, keeps track of where they are located on the disk.
7. What does it mean that Unix filenames are case-sensitive?

   Upper and lower case matter, so for example, "mydoc.txt" and "MyDoc.txt" are different files.
8. What is a root directory?

   The only directory that is not "contained" by another. The root (maybe better called trunk) of the directory tree.
9. How many root directories does a Unix system have? How many does Windows have?

   Unix: 1 Windows: 1 for each disk partition.

10. What is contained in the /bin and /usr/bin directories?
    Standard Unix commands, plus add-ons on some systems.
11. What is a subdirectory?
    A directory within another directory.
12. What is a home directory?
    A directory owned by a particular user, under which most or all of that users files are stored.
13. What is an absolute path name and how do we recognize one?
    The complete path from the root directory to a given file or directory name. It always starts with a / or ~.
14. What is the absolute path name of Sue's asg01.c in the tree diagram in this section?
    /home/sue/Asg1/asg01.c
15. Of what is the CWD a property?
    Each process.
16. Show a Unix command that prints the CWD of a shell process.
    pwd
17. Show a Unix command that sets the CWD of a shell process to /tmp.
    cd /tmp
18. Show a Unix command that sets the CWD of a shell process to our home directory?
    cd
19. What is a relative path name and how to we recognize one?
    A path name relative to CWD. It does not start with a / or ~.
20. Is a relative path name unique? Prove your answer with an example.
    No. There could be two directories called /home/joe/Asg01 and /home/sue/Asg01. Asg01 could refer to either of them depending on the CWD of the process.
21. How does Unix determine the absolute path name from a relative path name?
    By appending a '/' and the relative path name to CWD.
22. If the CWD of a process is /usr/local, what is the absolute path name of "bin/ape"?
    /usr/local/bin/ape
23. If the CWD of a process is /usr/local, what is the relative path name of /usr/local/lib/libxtend.a?
    lib/libxtend.a
24. If the CWD of a process is /usr/local, what is the relative path name of /usr/bin?
    ../bin
25. If the CWD of a process is /usr/local, what is the relative path name of /etc/motd?
    ../../etc/motd
26. Where does a new process get its initial CWD?
    It is inherited from the parent process.
27. Why should we avoid using absolute path names in programs and scripts?
    Absolute path names are not portable. When we copy the code to another computer, the absolute path names we used may not exist there, and we will have to make many changes to the code, which often results in regressions (new bugs).

28. Show a Unix command that lists the contents of the parent directory of CWD.

    ls ..

29. If the CWD of a process is /home/bob/Programs, what is the relative path name of /home/bob/Data/input1.txt?

    ../Data/input1.txt

30. How do we remove a file called "~sue" in the CWD?

    rm ./~sue

31. What are the three user categories that can be granted permissions on a file or directory?

    Individual owner, group owner, and everyone else

32. What does it mean to set execute permission on a file? On a directory?

    File: The file can be executed, assuming it is a program. Directory: Processes can make it their CWD.

33. Given the following ls -l output, who can do what to bootcamp.pdf?

```
-rw-r-----   1 joe    users    82118 Aug  2 09:47 bootcamp.pdf
```

    Joe can modify it, members of the group users can read it, and other users do not have access.

34. How would we allow users who are not in the owning group to read bootcamp.pdf?

    chmod o+r bootcamp.pdf

35. How would we allow members of the group to read and execute the program "simulation" and at the same time revoke all access to other users?

    chmod g+rx,o-rwx simulation

36. Show a Unix command that makes the directory "MyScripts" world writable.

    This should never be done!

37. Show a Unix command that changes the group ownership of the directory "Research" to the group "smithlab".

    chgrp smithlab Research

38. Assuming your primary group is "joe", show a Unix command that configures the directory Research form the previous question so that new files you create in it will be owned by "smithlab" instead of "joe"?

```
chmod g+s Research
```

3.10 Unix Commands and the Shell

3.10.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What types of commands have to be internal to the shell? Give one example and explain why it must be internal.

   Commands that alter the state of the shell process, such as **cd**. A child process cannot change CWD for its parent process.

2. How can you find a list of the basic Unix commands available on your system?

   ```
   ls /bin /usr/bin
   ```

3. How can you find out whether the **grep** command is internal or external, and where it is located?

   ```
   which grep
   type grep
   ```

4. What kind of suffering did computer users have to endure in order to read documentation before the Unix renaissance? How did Unix put an end to such suffering?

   They had to stand up and go get a printed manual. Unix introduced "online documentation" in the form of "man pages".

5. Show a Unix command that helps us learn about all the command-line flags available for the **tail** command.

   ```
   man tail
   ```

6. Show a Unix command that copies the file /tmp/sample.txt to the CWD.

   ```
   cp /tmp/sample.txt .
   ```

7. Show a Unix command that copies all files in /tmp whose names begin with "sample" and end with ".txt" to the CWD.

   ```
   cp /tmp/sample*.txt .
   ```

8. Show a Unix command that moves all the files in the CWD whose names end with ".py" to a subdirectory of the CWD called "Python".

```
mv *.py Python
```

9. Show a Unix command that creates another file name in the CWD called test-input.txt for the existing file ./Data/input.txt.

```
ln Data/input.txt test-input.txt
```

10. What is a hard link?

    A directory entry that points directly to the file content.

11. What is a symbolic link?

    A directory entry that points to another path name rather than the file content.

12. What do we get when we remove the path name to which a symbolic link points?

    A dangling link, which can no longer be used to access the file.

13. What limitations do hard links have that soft links do not have?

    Hard links cannot be used for directories and hard links to the same file must be in the same file system.

14. How do we create a new directory /home/joe/Data/Project1 if the Data directory does not exist and the CWD is /home/joe?

```
mkdir -p Data/Project1
mkdir -p ~joe/Data/Project1
mkdir -p /home/joe/Data/Project1
```

15. How do we remove the directory ./Data if it is empty? If it is not empty?

```
Empty:      rmdir Data
Not empty:  rm -r Data
```

16. Show a Unix command that tells us how much disk space is available in each file system.

```
df
```

17. Show a Unix command that tells us how much space is used by the directory ./Data.

```
du -sh Data
```

18. Show a sequence of Unix commands that change CWD to /tmp, then to /etc and then return to the original CWD.

```
pushd /tmp
cd /etc
popd
```

19. How do we exit the shell?

```
exit
```

20. Show a Unix command that tells us if there are carriage returns in graph.py.

```
vis graph.py
cat -v graph.py
```

21. Show a Unix command that displays the first 20 lines of output.txt.

```
head -n 20 output.txt
```

22. Show a Unix command that displays the last 20 lines of output.txt.

```
tail -n 20 output.txt
```

23. Show a Unix command that displays what has changed between analysis.c.old and analysis.c.

```
diff -u analysis.c.old analysis.c
```

24. Which text editor is available on all Unix systems?
    vi

25. Show a Unix command that tells us the name of the machine running our shell.

```
hostname
```

26. Show a Unix command to the remote server unixdev1.ceas.uwm.edu as user joe in order to run commands on it.

```
ssh joe@unixdev1.ceas.uwm.edu
```

27. Show a Unix command to change our local password.

```
passwd
```

28. How do we change our password for a Unix system that relies on LDAP or AD?

```
Usually, via a web portal to the LDAP/AD server.
```

29. Show a Unix command that clears the terminal display.

```
clear
```

30. Show a Unix command to reset the terminal mode to default settings.

```
reset
```

3.11 POSIX and Extensions

3.11.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is POSIX and why is it important?

   POSIX means the Portable Operating System standards based on UnIx. It provides a standard to which Unix systems conform, so that our programs, our scripts, and our knowledge of Unix commands will be as portable as possible.

2. What is an extension?

   An extension is a feature of a program that does not conform to a standard such as POSIX.

3. Does the use of extensions always prevent things from working on other Unix systems?

   No. Many programs that contain extensions are open source and can be easily installed on other systems.

3.12 Subshells

3.12.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. Show a single Unix command that runs pwd and produces the output "/etc", without changing the CWD of the shell process.

```
(cd /etc; pwd)
```

3.13 Redirection and Pipes

3.13.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. How does device independence simplify life for Unix users? Give an example.

   Input/output devices are treated exactly the same as ordinary files. For example, we can use the same Unix commands and programming language features to read a keyboard or mouse as we do a regular file.

2. Show an example Unix command that displays the input from a mouse as it is being moved or clicked.

   ```
   hexdump /dev/sysmouse
   ```

3. What are the standard streams associated with every Unix process? To what file or device are they connected by default?

   Standard input (default = terminal keyboard), standard output (default = terminal screen), and standard error (default = terminal screen).

4. Show a Unix command that saves the output of **ls -l** to a file called long-list.txt.

   ```
   ls -l > long-list.txt
   ```

5. Show a Unix command that appends the output of **ls -l /etc** to a file called long-list.txt.

   ```
   ls -l /etc >> long-list.txt
   ```

6. Show a Unix command that discards the normal output of **ls -l /etc** and shows the error messages on the terminal screen.

   ```
   ls -l /etc > /dev/null
   ```

7. Show a Bourne shell command that saves the output of **ls -al /etc** to output.txt and any error messages to errors.txt.

   ```
   ls -al /etc > output.txt 2> errors.txt
   ```

8. Show a C shell command that saves the output and errors of **ls -al /etc** to all-output.txt.

```
ls -al /etc >& all-output.txt
```

9. How does **more list.txt** differ from **more < list.txt**?

   In the first form, **more** receives the file name as an argument and creates a new stream to read from it. In the second form with redirection, the shell opens the file and connects the standard input stream of the **more** process to it. The **more** process doesn't even know the name of the file it is reading.

10. Show a Unix command that creates a 1 gigabyte file called `new-image` filled with 0 bytes.

```
dd if=/dev/zero of=new-image bs=1000000 count=1000
```

11. What are two major advantages of pipes over redirecting to a file and then reading it?

    - A pipe eliminates the need to create an intermediate file, which saves disk space and time.
      A pipe runs two processes at the same time, which is usually faster than running one after the other.

12. Show a Unix command that lists all the files in and under /etc, sorts them, and paginates the output.

```
find /etc | sort | more
```

13. What is a foreground process?

    A process that receives input from the terminal keyboard.

14. Which program in the following pipeline runs in the foreground?

```
shell-prompt: find /etc | sort | more
```

    more

15. What is a filter program?

    A program that can receive input from the standard input and send output to the standard output. Hence, it can be used within a pipeline.

16. What is the maximum number of commands allowed in a Unix pipeline?

    There is no limit. We are only limited by available memory to support all the processes.

17. Show a Unix command that prints a long listing of /usr/local/bin to the terminal and at the same time saves it to the file local-bin.txt.

```
ls -l /usr/local/bin | tee local-bin.txt
```

18. Do the same as above, but include any error messages in the file as well. Show the command for both C shell and Bourne shell.

```
C shell:        ls -l /usr/local/bin |& tee local-bin.txt
Bourne shell:   ls -l /usr/local/bin 2>&1 | tee local-bin.txt
```

19. Is it a good idea to feed files into a pipe using **cat**, rather than have the next command read them directly? Why or why not?

```
Example: Which command below is more efficient?
cat file.txt | sort | uniq
sort file.txt | uniq
```

    Using cat this way only increases resource use and slows down processing. It adds overhead for managing the pipe and does not benefit performance, because cat is not CPU-bound and therefore cannot utilize another core effectively.

3.14 Power Tools for Data Processing

3.14.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is a regular expression? Is it the same as a globbing pattern?

   A RE is an expression that represents a flexible pattern rather than a fixed string. RE patterns resemble globbing patterns, but they are not the same.

2. Show a Unix command that shows lines in analysis.c containing hard-coded floating point constants.

   ```
   grep '[+-]?[0-9]+\.[0-9]+' analysis.c
   ```

3. How can we speed up grep searches when searching for a fixed string rather than an RE pattern?

   Use **fgrep** or **grep --fixed-strings** instead of **grep**.

4. How can we use extended REs with grep?

   Use **egrep** or **grep --extended-regexp**.

5. How can we make the matched pattern visible in the grep output?

   Use the `--color` flag with grep, fgrep, or egrep.

6. Describe two major differences between grep and awk.

   1. Awk can process individual columns in a file. 2. Awk includes a powerful scripting language so that we can take virtually any action we want when matches are found.

7. How does awk compare to spreadsheet programs like LibreOffice Calc and MS Excel?

   Awk is mainly for processing data automatically, while spreadsheets are primarily interactive.

8. The /etc/group file contains colon-separated lines in the form groupname:password:groupid:members. Show an awk command that will print the groupid and members of the group "root".

```
awk -F : '$1 == "root" { print $3, $4 }' /etc/group
```

9. A GFF3 file contains tab-separated lines in the form "seqid source feature-type start end score strand phase attributes". The first attribute for an exon feature is the parent sequence ID. Write an awk script that reports the seqid, start, end, strand, and parent for each feature of type "exon". It should also report the number of exons and the number of genes. To test your script, download Mus_musculus.GRCm39.107.chromosome.1.gff3.gz from ensembl.org and then do the following:

```
gunzip Mus_musculus.GRCm39.107.chromosome.1.gff3.gz
awk -f your-script.awk Mus_musculus.GRCm39.107.chromosome.1.gff3
```

```
BEGIN {
    gene_count = exon_count = 0;
}

$3 == "exon" {
    # Separate attributes into an array
    split($9, attributes, ";");

    # Print location and feature ID
    printf("%s %s %s %s %s\n", $1, $4, $5, $7, attributes[1]);

    # Count this gene
    ++exon_count;
}
$3 == "gene" {
    ++gene_count;
}
END {
    printf("\nExons found = %d\n", exon_count);
    printf("Genes found = %d\n", gene_count);
}
```

10. Show a cut command roughly equivalent to the following awk command, which processes a tab-separated GFF3 file.

```
awk '{ print $1, $3, $4, $5 }' file.gff3
```

```
cut -f 1,3-5 file.gff3
```

11. Show a sed command that replaces all occurrences of "wolf" with "werewolf" in the file halloween-list.txt.

```
sed -i '' -e 's|wolf|were&|g' halloween-list.txt
```

12. Show a command to sort the following data by height. Show a separate command to sort by weight. The data are in params.txt.

```
ID  Height  Weight
1   34      10
2   40      14
3   29      9
4   28      11
```

```
height: sort -k 2 -n params.txt
weight: sort -k 3 -n params.txt
```

13. Show a Unix command that reads the file fox.txt, replaces the word "fox" with "toad" and converts all lower case letters to upper case, and stores the output in big-toad.txt.

```
sed -e 's|fox|toad|g' | tr '[:lower:]' '[:upper:]' > big-toad.txt.
```

14. Show a Unix command that lists and removes all the files whose names end in '.o' in and under ~/Programs.

```
find ~/Programs -name '*.o' -exec rm '{}' +
```

15. Why is the xargs command necessary?

    Unix has a limit on the length of a command, so we need a way to split up commands when processing thousands of arguments or more.

16. Show a Unix command that removes all the files with names ending in ".tmp" only in the CWD, assuming that there are too many of them to provide as arguments to one command. The user should not be prompted for each delete. (Check the **rm** man page if needed.)

```
find . -maxdepth 1 -name '*.tmp' | xargs rm -f

or

find . -maxdepth 1 -name '*.tmp' -exec rm -f '{}' +
```

17. Show a Unix command that processes all the files named 'input*' in the CWD, using as many cores as possible, through a command such as the following:

```
analyze --limit 5 input1 input2
```

```
find . -maxdepth 1 -name 'input*' | xargs --max-procs 0 analyze --limit 5
```

18. What is the most portable and flexible way to use xargs when the arguments it provides to the command must precede some of the fixed arguments?

    Write a simple shell script that take the arguments in the order provided by xargs and rearranges them for the needs of the command.

19. What is the major advantage of the **bc** calculator over common programming languages?

    It offers unlimited precision and range.

20. Show a bc expression that prints the value of the natural number, e.

    e(1)

21. Write a **bc** script that prints the following. Create the script with **nano sqrt.bc** and run it with **bc -l < sqrt.bc**.

```
sqrt(1) = 1.00000000000000000000
sqrt(2) = 1.41421356237309504880
sqrt(3) = 1.73205080756887729352
sqrt(4) = 2.00000000000000000000
sqrt(5) = 2.23606797749978969640
sqrt(6) = 2.44948974278317809819
sqrt(7) = 2.64575131106459059050
sqrt(8) = 2.82842712474619009760
sqrt(9) = 3.00000000000000000000
sqrt(10) = 3.16227766016837933199
```

```
#!/usr/bin/bc -l

for (c = 1; c <= 10; ++c)
{
    print "sqrt(", c, ") = ", sqrt(c), "\n"
}
quit
```

22. What are some advantages of archiving files in a tarball?
    1. Save disk space. 2. Reduces directory listings. 3. Processing one large file is faster than processing many small files.
23. Show a Unix command that creates a tarball called research.tar containing all the files in the directory ./Research.

```
tar -cvf research.tar Research
```

24. Show a Unix command that saves the output of **find /etc** to a compressed text file called `find-output.txt.bz2`.

```
find /etc | bzip2 > find-output.txt.bz2
```

25. Show a Unix command for viewing the contents of the compressed text file output.txt.gz, one page at a time.

```
zcat output.txt.gz | more
```

26. Show a Unix command that creates a tarball called research.txz containing all the files in the directory ./Research.

```
tar -Jcvf research.tar Research
```

27. What are **zip** and **unzip** primarily used for on Unix systems?
    Interoperability with Windows file archives.
28. Show a Unix command that reports the CPU time used by the command **awk -f script.awk input.tsv**.

```
time awk -f script.awk input.tsv
```

29. Show a Unix command that will help us determine which processes are using the most CPU time or memory.

```
top
```

30. How can we find out how to adjust the behavior of **top** while it is running?
    Press 'h' to see the help screen.
31. What kind of output from **top** might suggest that a process is I/O-bound? Why?
    Low CPU utilization. Processes do not use a CPU core while waiting for disk or other I/O.
32. Show a Unix command that continuously monitors total disk activity on a Unix system.

```
iostat 1
```

33. How can users who do not have access to an HPC cluster run things in parallel, in a more sophisticated way than possible with standard Unix tools such as xargs?
    Install GNU Parallel via the local package manager.

3.15 File Transfer

3.15.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What are three commands we can use in place of a web browser to download files? Which should we generally use in scripts that need to be portable? Why?

   Curl, fetch, and wget. Fetch is FreeBSD-specific, so we should generally use curl or wget for portable scripts.

2. Name three programs that can be used to transfer files between two systems.

   SFTP, FileZilla, scp, rsync.

3. Describe three advantages of **rsync** over other file transfer tools.

   1. Can be scripted to automate transfers. 2. Can quickly synchronize a destination after making changes to the source, without resending everything. 3. Can resume a failed transfer by rerunning the exact same command.

4. What is the meaning of a trailing '/' on the source directory?

   It tells **rsync** not to create the source directory under the destination directory, but copy the contents to it instead.

5. Show an **rsync** command that makes the directory `~/Data/Study1` on unixdev1.ceas.uwm.edu identical to `MyStudy` on the local machine.

   ```
   rsync -av --delete MyStudy/ unixdev1.ceas.uwm.edu:Data/Study1
   ```

6. Show an **rsync** command that makes the local directory `MyStudy` identical to `~/Data/Study1` on unixdev1.ceas.uwm.edu.

   ```
   rsync -av --delete unixdev1.ceas.uwm.edu:Data/Study1/ MyStudy
   ```

3.16 Environment Variables

3.16.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is the environment in Unix?

   A set of variables and values that are passed on to child processes.

2. What data types are available for environment variables?

   All environment variables are character strings.

3. Does a Unix process have any environment variables when it starts? If so, where do they come from?

   Yes, they are inherited from the parent process.

4. What is the purpose of the TERM environment variable? What kinds of programs make use of it?

   It indicates the type of terminal or terminal emulator in use. It is needed by programs such as editors, which need to send magic sequences to the terminal to move the cursor, change the color, etc.

5. What is the purpose of the PATH environment variable? What kinds of programs make use of it?

   It indicates where to look for external commands. It is used by programs that run other programs, like a shell or the **env** command.

6. Show how to set the environment variable TERM to the value "xterm" in

   (a) Bourne shell (sh)

   ```
   TERM=xterm
   export TERM
   ```

(b) Korn shell (ksh)

```
TERM=xterm
export TERM
```

(c) Bourne again shell (bash)

```
TERM=xterm
export TERM
```

(d) C shell (csh)

```
setenv TERM xterm
```

(e) T-shell (tcsh)

```
setenv TERM xterm
```

7. Show a Unix command that runs **ls** with the LSCOLORS environment variable set to "CxFxCxDxBxegedaBaGaCaD". You may not change the LSCOLORS variable for the current shell process.

```
env LSCOLORS=CxFxCxDxBxegedaBaGaCaD ls
```

3.17 Shell Variables

3.17.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is the difference between shell variables and environment variables?

   Shell variables are used only by the shell and not inherited by child processes.

2. Show how to set the shell prompt to "Unixdev1: " in:

   (a) Bourne shell

   ```
   PS1="Unixdev1: "
   ```

   (b) C shell

   ```
   set prompt="Unixdev1: "
   ```

3. How can you view a list of all current shell variables and their values?

   ```
   set
   ```

3.18 Process Control

3.18.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is the difference between a foreground process and a background process? How many of each can be running under a given shell process?

   A foreground process receives input from the keyboard. We can have only one foreground process under each shell process, but any number of background processes.

2. How do we keep the output of many background processes from mixing together?
   By redirecting the output of all background processes.
3. Show a Unix command that terminates process 8210 if **kill 8210** has no effect.

```
kill -9 8210
kill -KILL 8210
```

4. What can we do if we have many processes running a program called **fastqc** and we want to terminate all of them?

```
pkill fastqc
```

5. What is the easiest way to terminate the foreground process?
   Press Ctrl+c
6. How can we temporarily stop the foreground process, run **ls** under the same shell, and then continue the previous process.

```
Ctrl+z
ls
fg
```

7. How can we pause output from processes in a terminal, inspect it, and then allow it to continue?

```
Ctrl+s
Ctrl+q
```

8. How can we stop the current foreground process and resume it as a background process?

```
Ctrl+z
bg
```

9. Show a Unix command that runs ./analysis so that its process uses less CPU time than other processes.

```
nice ./analysis
```

10. Show a Unix command that runs ./analysis so that it will continue running even after we log out.

```
nohup ./analysis >& output.txt &       # C shell
nohup ./analysis > output.txt 2>&1 &   # Bourne shell
```

3.19 Remote Graphics

3.19.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is X11?

   X11 is a networked graphics API used by Unix systems that allows programs to display graphics both on the local display and displays of other computers over a network.

2. Does Apple's macOS use X11? Explain.

   Not by default. We must install the free XQuartz software to add X11 capabilities to macOS.

3. What must be installed at minimum on a remote computer to allow X11 client programs to run there and display graphics on the X11 display in front of you?

   xauth

4. Show a Unix command that logs into unixdev1.ceas.uwm.edu and allows us to run remote graphical programs.

   ```
   ssh -X username@unixdev1.ceas.uwm.edu

   or

   ssh -Y username@unixdev1.ceas.uwm.edu
   ```

5. Show a Unix command that logs into unixdev1.ceas.uwm.edu and allows us to run remote graphical programs over a slow network.

   ```
   ssh -XC username@unixdev1.ceas.uwm.edu

   or

   ssh -YC username@unixdev1.ceas.uwm.edu
   ```

6. Why is setting ForwardX11Trusted not generally a good idea?

   It implicitly trusts all remote hosts that eny user connects to, which could expose the system to password capture and other deceptive attacks.

7. What packages are needed on a Cygwin setup to enable X11?

   xinit and xhost

Chapter 4

Unix Shell Scripting

4.1 What is a Shell Script?

4.1.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is a shell script?

   A shell script is a file containing a sequence of Unix commands exactly as they would be typed interactively via the keyboard.

2. Compare and contrast Unix commands with subprogram calls in a C or Java program.

   Both contain a name followed by zero or more arguments. The only difference is in how the arguments are separated from the name and each other. Subprogram calls use parenthesis and commas, while Unix commands use white space.

3. What is the difference between a script and a "real" program?

   Scripts to not perform much, if any, computation on their own. They simply automate the execution of other programs that do the computation.

4. Is a scripting language a good choice for performing matrix multiplication? Why or why not?

   No. Scripting languages are interpreted and therefore slow. Computationally intensive code like matrix manipulation should be done in a compiled language.

4.2 Why Write Shell Scripts?

4.2.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. Describe three reasons for writing shell scripts instead of running commands from the keyboard.

   - Efficiency: Don't waste time typing the same sequence of commands repeatedly
   - Accuracy: Retyping the same sequence of commands is error-prone
   - Documentation: A script documents an analysis in perfect detail

2. What feature of Unix makes scripting so easy to implement and use? Explain.

   Device independence. Since the keyboard looks like an input file to Unix programs, we can put the same Unix commands in a file that we would type at the shell prompt, and they will work exactly the same way.

3. When should we run commands at the shell prompt and when should we put them in a script?

   If we are sure we won't need to run them again, we can run them at the shell prompt. If we might have to run them again, we should put them in a script.

4. What are three advantages of a script over a document explaining the commands to run?
   - People can run a script. They cannot run a document.
   - In creating the document, we have to type in every command twice; Once at the shell prompt and again in the document.
   - We have to remember to update the document every time we make a change to the commands we use. Nobody is this perfectly disciplined.
5. What type of flag arguments should we use in scripts? Why?

   Use the long-flags, such as `--compress` rather than `-z` whenever possible. This will save us and others the time of having to look up their meaning.
6. What is the advantage of using an integrated development environment to write scripts, rather than a simple editor like nano?

   We can test changes to the script with one keystroke and we don't lose our place in the file.
7. What is the advantage of Unix scripting languages over others such as Visual Basic or AppleScript? What is the disadvantage?

   Unix scripts can be run on any OS, including macOS (which is Unix) and Windows with Cygwin or WSL. There really is no disadvantage, beyond the minor effort required to install Cygwin.

4.3 Which Shell?

4.3.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What are the two families of shell programs?

   Bourne shell and C shell

2. State one advantage of Bourne shell for scripting and one advantage of C shell.

   Bourne shell has subprograms. C shell has a cleaner and more intuitive syntax, similar to C.

3. What is the advantage of using POSIX Bourne shell for scripting rather than an extended shell such as **bash** or **dash**?

   All Unix systems have a POSIX-compatible Bourne shell, so the scripts will be highly portable.

4. What is the disadvantage of using POSIX Bourne shell for scripting rather than an extended shell such as **bash** or **dash**?

   Not much. There are some additional scripting features, but they don't make scripting much easier. Their main advantages are in interactive use.

4.4 Writing and Running Shell Scripts

4.4.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. Is it wise to write Unix shell scripts in a Windows editor and then upload them to a Unix system? Why or why not?

   No. The carriage returns in Windows text files may cause problems in Unix scripts. It is also a very inefficient work flow.

2. How can we keep very long commands tidy in a script?

   Use the continuation character "\" at the end of a line and continue the command on the next line.

3. What rules does Unix enforce regarding file name extensions on shell scripts?

   None

4. What problems might arise if we write a **bash** script and give it a ".sh" file name extension?

   Someone might be misled into thinking the script adheres to POSIX Bourne shell syntax, and the script won't work with **sh** on their system.

5. Why are comments important in shell scripts?

   Same reason as any other language: Some commands in the script may be difficult to understand and require some explanation.

6. Show three ways to run a Bourne shell script called `analysis.sh` and any requirements they entail.

   - ```
     # Only requires that the script is Bourne shell compatible
     sh < analysis.sh
     ```
   - ```
     # Only requires that the script is Bourne shell compatible
     sh < analysis.sh
     ```
   - ```
     # Requires a shebang line #!/bin/sh and execute permissions on analyze.sh
     ./analyze.sh
     ```

7. What shebang line should be used for Bourne shell scripts? For Bourne again shell scripts? For Python scripts? Explain.

   - Bourne shell: #!/bin/sh. Bourne shell is /bin/sh on all Unix systems.
   - Bourne again shell: #!/usr/bin/env bash. Bash may be installed in different directories on different systems, or we may want to use a newer version than /bin/bash on RHEL.
   - Python: #!/usr/bin/env python. Python may be installed in different directories on different systems.

8. How to we make a shell script exit immediately when any command fails?

   Add -e to the shebang line, unless using /usr/bin/env. In that case, run **set -e** in the script for Bourne shell family.

9. What happens if we run a script as an argument to a shell command that does not match the intended shell, such as **csh analysis.sh**?

   It may or may not work, depending on whether the script uses any features of the intended shell that are not supported by the shell explicitly used.

10. What are the advantages of using an IDE instead of a simple text editor like nano?

    It greatly speeds up script development by allowing us to run the script without leaving the editor or using a separate shell window. IDEs also provide features to make coding easier.

4.5 Sourcing Scripts
     ```

4.5.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What happens when we run a script by simply typing its name at the shell prompt?

   A new shell process is started to run the script, using the shebang line contained in the script to indicate which shell to use.

2. How does sourcing a script differ from running it the usual way?

   Sourcing causes the current shell process to execute the commands in the script instead of giving the task to a child process.

3. Under what circumstances would we want to source a script rather than run it under a child shell process?

   When we want the commands in the script to affect the environment or shell variables of the current process, which is impossible if a child process runs the script.

4.6 Shell Start-up Scripts

4.6.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What are startup scripts?

   Startup scripts are shell scripts that are automatically sourced when a new shell process is started.

2. Are startup scripts sourced by shell processes running other scripts?

   It depends. Bourne shell family shells to not source startup scripts by default. C shell family scripts do, but this can be disabled by adding `-f` to the shebang line.

3. What are startup scripts used for?

   Startup scripts are used to personalize our shell environment, such as adding components to PATH or customizing the shell prompt.

4.7 String Constants and Terminal Output

4.7.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. How does the Unix shell interpret the character sequence `firstname`? How would it be interpreted by most general purpose programming languages?

   The shell sees it as a string constant, while most languages would see it as a variable name.

2. Show three ways to make the shell see the character sequence `Alfred E. Neumann` as a single string rather than three strings. Note that there are two spaces after the period.

   "Alfred E. Neumann" 'Alfred E. Neumann' Alfred\ E.\ \ Neumann

3. What is the output of the following statement? How do we make it interpret the \n as a newline?

```
#!/bin/sh -e

printf \$15*3=\$45\n
```

```
$15*3=$45n
```

```
#!/bin/sh -e

printf "\$15*3=\$45\n"
printf '\$15*3=\$45\n'
printf \$15*3=\$45\\n
```

4. What are some of the problems with the **echo** command? What is the solution?

   The echo command is not standardized. Use printf instead.

5. Show a **printf** statement that prints the number 32,767, right-justified in a field of 10 columns.

   printf '%10s\n' '32,767'

6. Show a **printf** statement that prints the error message "Error: Invalid data found in column 2 of input.".

   printf "Error: Invalid data found in column 2 of input.\n" >> /dev/stderr

4.8 Shell and Environment Variables

4.8.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is the convention for naming shell variables to help distinguish them from environment variables?

   Shell variables use all lower case and environment variables use all upper case.

2. Can we use any name we want for a shell variable in our script?

   No, some variable names are reserved for special purposes, such as `prompt` or `PS1`.

3. What rules do shell and environment variable names need to follow?

   '[A-Za-z_][A-Za-z0-9_]*'

4. Show how to assign the value "Alfred E. Neumann" to the shell variable full_name in Bourne shell and in C shell.

   Bourne: first_name='Alfred E. Neumann' C shell: set first_name = 'Alfred E. Neumann'

5. How do we declare a shell variable? Explain.

   We don't. Variables are created when assigned a value for the first time. Since all shell variables are character strings, there is no need to assign a type.

6. Are shell scripts a good choice for performing numeric computations? Why or why not?

   No. All variables are stored as character strings, which makes numeric computations very slow. Numeric capabilities are also very limited.

7. What is the relationship between a given shell variable and an environment variable with the same name? Explain.

   In the Bourne shell family, they always have the same value, since environment variables are set by assigning a shell variable and then exporting it. In the C shell family, they can be separate since shell variables are set with the **set** command and environment variables with the **setenv** command.

8. Are all C shell variable independent of environment variables? Use an example to clarify.

   No. The shell variable `path` is linked to the environment variable `PATH`.

9. Show the output of the following script:

```
#!/bin/sh -e

name="Wile E. Coyote"

printf "$name\n"
printf "name\n"
printf '$name\n'
```

```
Wile E. Coyote
name
$name
```

10. What is the output of the following script? What do you think is the intended output and how can we make it happen?

```
#!/bin/sh -e

file_size=200

printf "File size is $file_sizeMB.\n"
```

    Output is "File size is MB."

```
printf "File size is ${file_size}MB.\n"
```

11. What is the danger in the following script? Alter the script to eliminate the risk.

```
#!/bin/sh -e

printf "The first 20 lines of file.txt are:\n"
head -n 20 file.txt
printf "The last 20 lines of file.txt are:\n"
tail -n 20 file.txt
```

    The hard-coded file name and line count will need to be updated in multiple places, which is an error-prone process. We can use variables instead:

```
#!/bin/sh -e

readonly file_name="file.txt"
readonly line_count=20

printf "The first $line_count lines of $file_name are:\n"
head -n $line_count $file_name
printf "The last $line_count lines of $file_name are:\n"
tail -n $line_count $file_name
```

12. Write a shell script that prints the following, using a single **printf** command and using **wc -l** to find the number of lines in the file. Exact white space is not important.

```
input1.txt contains 3258 lines.
```

```
#!/bin/sh -e

readonly file_name="input1.txt"

printf "$file_name contains `wc -l $file_name` lines.\n"
```

4.9 Hard and Soft Quotes

4.9.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What shell constructs are expanded when found inside hard quotes?

   Only history events, starting with a '!'.

2. What shell constructs are expanded when found inside hard quotes?

   History events, starting with '!', variable references, starting with '$', and output capture (`` or $()).

3. What is the output of the following script if it is run in /home/joe?

```
#!/bin/sh -e

cwd=$(pwd)
printf "The CWD is $cwd.\n"
string='The CWD is $cwd.\n'
printf "$string"
```

```
The CWD is /home/joe.
The CWD is $cwd.
```

4. How do we print a single quote character in a shell command?

   Either escape it (precede it with a \) or enclose the string in double quotes.

5. How do we concatenate two strings in a shell command?

   Simply place them next to each other with nothing between them.

4.10 User Input

4.10.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. Write a shell script that lists the files in the CWD, asks the user for the name of a file, a source string, and a replacement string, and then shows the file content with all occurrences of the source replaced by the replacement.

```
shell-prompt: cat fox.txt
The quick brown fox jumped over the lazy dog.

shell-prompt: ./replace.sh
Documents          R                 fox.txt           stringy
Downloads          igv               stringy.c
File name? fox.txt
Source string? fox
Replacement string? tortoise
The quick brown tortoise jumped over the lazy dog.
```

```
#!/bin/sh -e

ls
printf "File name? "
read filename
printf "Source string? "
read source
printf "Replacement string? "
read replacement
sed -e "s|$source|$replacement|g" $filename
```

4.11 Conditional Execution

4.11.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is the exit status of a command that succeeds? One that fails?

   0 on success, a variety of non-zero error codes on failure.

2. What variables contain the exit status of the most recent command in Bourne shell and C shell?

   Bourne shell: $? C shell: $status

3. What is the meaning of **[** in a shell script and how is it used?

   It is an external command, equivalent to **test**. It is often used to convert Boolean expressions or file tests to the exit status required by the **if-then-else** in Bourne shell scripts.

4. Write a shell script that lists the files in the CWD, asks the user for the name of a file, tests for the existence of the file, and issues an error message if it does not exist. If the file does exist, the script then asks for a source string and a replacement string, verifying that the source string is not empty, and then shows the file content with all occurrences of the source replaced by the replacement. The script should exit with status 65 (EX_DATAERR) if any bad input is received.

```
shell-prompt: cat fox.txt
The quick brown fox jumped over the lazy dog.

shell-prompt: ./replace.sh
Documents          R                  fox.txt            stringy
Downloads          igv                stringy.c
File name? fxo.txt
File fxo.txt does not exist.
shell-prompt: echo $?
65

shell-prompt: ./replace.sh
Documents          R                  fox.txt            stringy
Downloads          igv                stringy.c
File name? fox.txt
Source string?
Source string cannot be empty.
shell-prompt: echo $?
65

shell-prompt: ./replace.sh
Documents          R                  fox.txt            stringy
Downloads          igv                stringy.c
File name? fox.txt
Source string? fox
Replacement string? tortoise
The quick brown tortoise jumped over the lazy dog.
shell-prompt: echo $?
0
```

```
#!/bin/sh -e

EX_DATAERR=65

ls
printf "File name? "
read filename
if [ -e "$filename" ]; then
    printf "Source string? "
    read source
    if [ 0"$source" != 0 ]; then
        printf "Replacement string? "
        read replacement
        sed -e "s|$source|$replacement|g" $filename
    else
        printf "Source string must not be empty.\n"
        exit $EX_DATAERR
    fi
else
    printf "$filename does not exist.\n"
    exit $EX_DATAERR
fi
```

```
#!/bin/csh -ef

set EX_DATAERR=65
```

```
ls
printf "File name? "
set filename="$<"
if ( -e "$filename" ) then
    printf "Source string? "
    set source="$<"
    if ( "$source" != "" ) then
        printf "Replacement string? "
        set replacement="$<"
        sed -e "s|$source|$replacement|g" $filename
    else
        printf "Source string must not be empty.\n"
        exit $EX_DATAERR
    endif
else
    printf "$filename does not exist.\n"
    exit $EX_DATAERR
endif
```

5. Modify the previous script so that it reports an error if the replacement string is empty or the same as the source. Use a conditional operator to check both conditions in one **if-then-else** command.

```
shell-prompt: ./replace.sh
Documents          R                  fox.txt            stringy
Downloads          igv                stringy.c
File name? fox.txt
Source string? fox
Replacement string?
Replacement must not be empty or the same as source.
shell-prompt: echo $?
65

shell-prompt: ./replace.sh
Documents          R                  fox.txt            stringy
Downloads          igv                stringy.c
File name? fox.txt
Source string? fox
Replacement string? fox
Replacement must not be empty or the same as source.
shell-prompt: echo $?
65
```

```
#!/bin/sh -e

EX_DATAERR=65

ls
printf "File name? "
read filename
if [ -e "$filename" ]; then
    printf "Source string? "
    read source
    if [ 0"$source" != 0 ]; then
        printf "Replacement string? "
        read replacement
        if [ 0"$replacement" = 0 ] || [ 0"$replacement" = 0"$source" ]; then
            printf "Replacement must not be empty or the same as source.\n"
        else
            sed -e "s|$source|$replacement|g" $filename
        fi
```

```
    else
        printf "Source string must not be empty or the same as replacement.\n"
        exit $EX_DATAERR
    fi
else
    printf "$filename does not exist.\n"
    exit $EX_DATAERR
fi
```

```
#!/bin/csh -ef

set EX_DATAERR=65

ls
printf "File name? "
set filename="$<"
if ( -e "$filename" ) then
    printf "Source string? "
    set source="$<"
    if ( "$source" != "" ) then
        printf "Replacement string? "
        set replacement="$<"
        if ( "$replacement" == "" || "$replacement" == "$source" ) then
            printf "Replacement must not be empty or the same as source.\n"
        else
            sed -e "s|$source|$replacement|g" $filename
        endif
    else
        printf "Source string must not be empty.\n"
        exit $EX_DATAERR
    endif
else
    printf "$filename does not exist.\n"
    exit $EX_DATAERR
endif
```

6. Write a shell script that asks the user for a directory name and the name of an archive to create from it, checks the file name extension on the archive name using a **switch/case**, and creates a tarball with the appropriate compression. The script should report an error and exit with status 65 if an invalid file name extension is used for the archive name, or if the directory name entered does not exist or is not a directory.

```
shell-prompt: ./case.sh
Coral               Qemu                case.sh             scripts
Directory to archive? Qem
Qem is not an existing directory.

shell-prompt: ./case.sh
Coral               Qemu                case.sh             scripts
Directory to archive? case.sh
case.sh is not an existing directory.

shell-prompt: ./case.sh
Coral               Qemu                case.sh             scripts
Directory to archive? Qemu
Archive name? qemu.ta
Invalid archive name: qemu.ta

shell-prompt: ./case.sh
Coral               Qemu                case.sh             scripts
Directory to archive? Qemu
Archive name? qemu.txz
```

```
a Qemu
a Qemu/FreeBSD-13.0-RELEASE-riscv-riscv64.raw
```

Extension	Tool
tar	No compression
tar.gz or tgz	gzip
tar.bz2 or tbz	bzip2
tar.xz or txz	xz

Table 4.1: Compression tool for each filename extension

In Bourne shell, the file name extension can be extracted from a shell variable as follows:

```
extension=${filename##*.}   # Strip off everything to the last '.'
```

In C shell:

```
set extension=${archive:e}  # Extract filename extension
```

```
#!/bin/sh -e

ls
printf "Directory to archive? "
read dir
if [ ! -d $dir ]; then
    printf "$dir is not an existing directory.\n"
    exit 65
fi

printf "Archive name? "
read archive
extension=${archive##*.}

case $extension in
tar)
    tar -cvf $archive $dir
    ;;

tar.gz|tgz)
    tar -zcvf $archive $dir
    ;;

tar.bz2|tbz)
    tar -jcvf $archive $dir
    ;;

tar.xz|txz)
    tar -Jcvf $archive $dir
    ;;

*)
    printf "Invalid archive name: $archive\n"
    ;;

esac
```

```
#!/bin/csh -ef

ls
```

```
printf "Directory to archive? "
set dir="$<"
if ( ! -d $dir ) then
    printf "$dir is not an existing directory.\n"
    exit 65
endif

printf "Archive name? "
set archive="$<"
set extension=${archive:e}

switch($extension)
case    tar:
    tar -cvf $archive $dir
    breaksw

case    tar.gz:
case    tgz:
    tar -zcvf $archive $dir
    breaksw

case    tar.bz2:
case    tbz:
    tar -jcvf $archive $dir
    breaksw

case    tar.xz:
case    txz:
    tar -Jcvf $archive $dir
    breaksw

default:
    printf "Invalid archive name: $archive\n"
    exit 65

endsw
```

7. Write a shell script that does the following in sequence:

   (a) Check for the existence of Homo_sapiens.GRCh38.107.chromosome.1.gff3 in the CWD. If it is not present, download the gzipped file using **curl** from http://ftp.ensembl.org/pub/release-107/gff3/homo_sapiens/ and decompress it.

   (b) Display the first 5 lines of the file that do not begin with '#', so the user can see the format of an entry. Hint: See the **grep** man page for an option to select lines that *do not* match the given pattern.

   (c) Ask the user which column to search, and then display all unique values in that column. Hint: Use **grep** again to filter out lines beginning with '#', run them through **cut** or **awk** to select just the desired column, and run the output through **sort** and **uniq** or just **sort** with the appropriate flags.

   (d) Ask the user for a search key, and display all the lines in the file not beginning with '#' that *contain* the given key in the given column. Hint: See the awk '~' operator and use `-v` to set `column` and `key` variables to use in the awk pattern.

```
shell-prompt: ./gff.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 4111k  100 4111k    0     0   550k      0  0:00:07  0:00:07 --:--:--  589k

1       GRCh38  chromosome      1       248956422       .       .       .       ID= ↩
    chromosome:1;Alias=CM000663.2,chr1,NC_000001.11
1       .       biological_region       10469   11240   1.3e+03 .       .       ↩
    external_name=oe %3D 0.79;logic_name=cpg
```

```
1       .       biological_region       10650   10657   0.999   +       .       ↩
    logic_name=eponine
1       .       biological_region       10655   10657   0.999   -       .       ↩
    logic_name=eponine
1       .       biological_region       10678   10687   0.999   +       .       ↩
    logic_name=eponine

Column to search? 3
CDS
biological_region
chromosome
exon
five_prime_UTR
gene
lnc_RNA
mRNA
miRNA
ncRNA
ncRNA_gene
pseudogene
pseudogenic_transcript
rRNA
scRNA
snRNA
snoRNA
three_prime_UTR
unconfirmed_transcript

Search key? UTR
1       havana  five_prime_UTR  65419   65433   .       +       .       Parent= ↩
    transcript:ENST00000641515
1       havana  five_prime_UTR  65520   65564   .       +       .       Parent= ↩
    transcript:ENST00000641515
1       havana  three_prime_UTR 70009   71585   .       +       .       Parent= ↩
    transcript:ENST00000641515
1       havana  five_prime_UTR  923923  924431  .       +       .       Parent= ↩
    transcript:ENST00000616016
1       havana  three_prime_UTR 944154  944574  .       +       .       Parent= ↩
    transcript:ENST00000616016
...
```

```
#!/bin/sh -e

# Download file if it we don't already have it
file=Homo_sapiens.GRCh38.107.chromosome.1.gff3
site=http://ftp.ensembl.org/pub/release-107/gff3/homo_sapiens
if [ ! -e $file ]; then
    curl -O $site/$file.gz
    gunzip $file.gz
fi

# Show file format
grep -v '^#' $file | head -n 5

printf "Column to search? "
read col

# Show all unique values in the given column
grep -v '^#' $file | cut -f $col | sort -u

printf "Search key? "
read key
```

```
# Show all lines containing key in the given column
awk -v col=$col -v key=$key '$col ~ key { print $0 }' $file | more
```

4.12 Loops

4.13 Generalizing Your Code

4.13.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. Modify the following shell script so that it takes the starting and ending values of the loop as user input rather than hard-coding them.

```
#!/bin/sh -e

for c in $(seq 1 10); do
    c_squared=$(($c * $c))
    printf "%s^2 = %s\n" $c $c_squared
done
```

```
shell-prompt: ./squares.sh
First value to square? 3
Last value to square? 9
3^2 = 9
4^2 = 16
5^2 = 25
```

```
6^2 = 36
7^2 = 49
8^2 = 64
9^2 = 81
```

```
#!/bin/sh -e

printf "First value to square? "
read first
printf "Last value to square? "
read last
for c in $(seq $first $last); do
    c_squared=$(($c * $c))
    printf "%s^2 = %s\n" $c $c_squared
done
```

2. Repeat the above exercise, but use command-line arguments instead of user input.

```
shell-prompt: ./squares.sh
Usage: ./squares.sh first-value last-value

shell-prompt: ./squares.sh 4 10
4^2 = 16
5^2 = 25
6^2 = 36
7^2 = 49
8^2 = 64
9^2 = 81
10^2 = 100
```

```
#!/bin/sh -e

if [ $# != 2 ]; then
    printf "Usage: $0 first-value last-value\n" >> /dev/stderr
    exit 64     # EX_USAGE, from sysexits.h
fi

first=$1
last=$2
for c in $(seq $first $last); do
    c_squared=$(($c * $c))
    printf "%s^2 = %s\n" $c $c_squared
done
```

4.14 Pitfalls and Best Practices

4.14.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What can we infer about the hardware on which our script is running based on the name of the operating system?

   Absolutely nothing. Most Unix systems run on multiple hardware platforms, so making any assumptions about the hardware based on the operating system is foolish.

2. How should a script decide what compiler to use to build programs?

   It shouldn't. This decision should be left to the user.

3. How should a script go about determining what type of CPU your system uses?

   Using a tool that directly reports the CPU type, not be inferring it from other information that we happen to know.

4.15 Script Debugging

4.15.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is execute tracing?

   A feature of Unix shells causing them to echo fully expanded commands to the standard error immediately before executing them.

2. How do we enable execute tracing for an entire script?

   Add the `-x` flag to the shebang line or the shell command starting the script.

3. How do we enable execute tracing for just a few commands in a script?

   Bourne shell: **set -x** to enable, **set +x** to disable. C shell: **set echo** to enable, **unset echo** to disable.

4.16 Functions, Child Scripts, and Aliases

4.16.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is the advantage of using a Bourne shell function as opposed to running a separate script?

   Convenience. A separate script must be placed in a directory in our PATH in order to be found. A function is inherently known to the script that contains it.

2. What are the advantages of using a separate script, as opposed to a Bourne shell function?

   - Separate scripts run under a separate child process, so they have their own local variables and cannot cause side-effects on the caller (unless run using **source** or **.**).
   - Using separate scripts encourages us to generalize them so they can be used by multiple other scripts. This helps avoid creation of redundant code and keeps our script collection more modular.

3. Show how to create an alias called **rls** which runs **find . -type f**.

```
# Bourne shell
alias rls='find . -type f'

# C shell
alias rls find . -type f
```

4.17 Here Documents

4.17.1 Practice

Instructions

1. Make sure you are using the latest version of this document.
2. Carefully read one section of this document and casually read other material (such as corresponding sections in a textbook, if one exists) if needed.
3. Try to answer the questions from that section. If you do not remember the answer, review the section to find it.
4. Write the answer in your own words. Do not copy and paste. Verbalizing answers in your own words helps your memory and understanding. Copying does not, and demonstrates a lack of interest in learning.
5. Check the answer key to make sure your answer is correct and complete.

   DO NOT LOOK AT THE ANSWER KEY BEFORE ANSWERING QUESTIONS TO THE VERY BEST OF YOUR ABILITY. In doing so, you would only cheat yourself out of an opportunity to learn and prepare for the quizzes and exams.

Important notes:

- Show all your work. This will improve your understanding and ensure full credit for the homework.
- The practice problems are designed to make you think about the topic, starting from basic concepts and progressing through real problem solving.
- Try to verify your own results. In the working world, no one will be checking your work. It will be entirely up to you to ensure that it is done right the first time.
- Start as early as possible to get your mind chewing on the questions, and do a little at a time. Using this approach, many answers will come to you seemingly without effort, while you're showering, walking the dog, etc.

---

1. What is a heredoc?

   A heredoc is text embedded in a script that is read using the redirection symbol <<. It can be used as input to any Unix command.

2. Can a heredoc contain anything besides literal text?

   Yes, it can contain shell and environment variable references and output capture.

4.18 Perl, Python, and other Scripting Languages

4.19 Scripting an Analysis Pipeline