COLORADO STATE UNIVERSITY GLOBAL

CSC507: Foundations of Operating Systems

Credit Hours: 3

Contact Hours: This is a 3-credit course, offered in accelerated format. This means that 16 weeks of material is covered in 8 weeks. The exact number of hours per week that you can expect to spend on each course will vary based upon the weekly coursework, as well as your study style and preferences. You should plan to spend 14-20 hours per week in each course reading material, interacting on the discussion boards, writing papers, completing projects, and doing research.

Faculty Information: Faculty contact information and office hours can be found on the faculty profile page.

Course Description and Outcomes

Course Description:

In this graduate course, students will develop a foundational knowledge in operating system concepts. Students will gain a detailed understanding of appropriate operating system constructs that involve OS abstractions and mechanisms. Students will also gain understanding of the constructs of multithreading and resource management in computer systems.

Course Overview:

This course will introduce you to the ins and outs of Computer Operating Systems. You will learn how and why operating systems evolved from the simple batch systems in the early days, to the advanced and varied systems we use today. You will understand the essential elements of operating systems, including processors, memory, and file management. This course will use both theory and practical application of the information learned.

Course Learning Outcomes:

1. Identify the resources and factors that affect computer system performance.
2. Implement techniques to improve parallelism and latency in an application.
3. Contrast the kernel and user mode in developing key approaches to operating system design and implementation.
4. Analyze processes that can be utilized to improve the operating system environment during execution.
5. Use system commands to manage I/O systems and file systems in an operating system environment.

Participation & Attendance

Prompt and consistent attendance in your online courses is essential for your success at CSU-Global Campus. Failure to verify your attendance within the first 7 days of this course may result in your withdrawal. If for some reason you would like to drop a course, please contact your advisor.

Online classes have deadlines, assignments, and participation requirements just like on-campus classes. Budget your time carefully and keep an open line of communication with your instructor. If you are having technical problems, problems with your assignments, or other problems that are impeding your progress, let your instructor know as soon as possible.

Course Materials

Required:

Stallings, W. (2018). *Operating systems: Internals and design principles* (9th ed.). Hoboken, NJ: Pearson Education, Inc.
ISBNs: 9780134670959, 0134670957, 9780134700113, 0134700112

NOTE: *All non-textbook required readings and materials necessary to complete assignments, discussions, and/or supplemental or required exercises are provided within the course itself. Please read through each course module carefully.*

Course Schedule

Due Dates

The Academic Week at CSU-Global begins on Monday and ends the following Sunday.

- **Discussion Boards:** The original post must be completed by Thursday at 11:59 p.m. MT and Peer Responses posted by Sunday 11:59 p.m. MT. Late posts may not be awarded points.
- **Critical Thinking:** Assignments are due Sunday at 11:59 p.m. MT.

Weekly Reading and Assignment Details

Module 1

Readings

- Chapter 1 in *Operating systems: Internals and design principles*
- Chapter 2 in *Operating systems: Internals and design principles*
- Denning, P. J. (2016). Fifty years of operating systems. *Communications of the ACM, 59*(3).
- Denning, P. (2015). Perspectives on OS foundations. In *SOSP '15: SOSP History Day 2015.*

Discussion (25 points)

Critical Thinking (75 points)

Option 1: Compare and contrast structural elements of computers.

Determine the specifications of the four main structural elements of your computer: processor, main memory, I/O modules, and system bus. Find this information for the other major type of computer OS, i.e. if your computer's OS is Windows, obtain this information for a Mac, or vice versa. You may need to use a library computer for this. Write a summary of your findings, comparing the specifications of the systems, along with your expectations of differences in performance and capacity of the computers running these systems.

Your paper should be 2-3 pages in length and contain at least four references. Use the CSU Global library to find additional appropriate references beyond the references given in the modules. Make sure it is in APA Format. A helpful resource for APA formatting can be found at the CSU Global Guide to Writing and APA.

Option 2: Compare and contrast structural elements of mobile devices.

Determine the specifications of the four main structural elements of your mobile device: processor, main memory, I/O modules, and system bus. Find this information for the other main type of mobile device OS, i.e. if your mobile phone runs on Android OS, obtain this information for an iPhone, or vice versa. You may need to visit a mobile phone store to get this information for a device with another OS, or use a friend or relative’s phone. Write a summary of your findings, comparing the specifications of the systems, along with your expectations of differences in performance and capacity of the devices running these systems.

Your paper should be 2-3 pages and contain at least four references. Use the CSU Global library to find additional appropriate references beyond the references given in the modules. Make sure it is in APA Format. A helpful resource for APA formatting can be found at the CSU Global Guide to Writing and APA.

Module 2

Readings

- Chapter 3 in *Operating systems: Internals and design principles*
- Chapter 4 in *Operating systems: Internals and design principles*
- Iwasaki, S., Amer, A., Taura, K., & Balaji, P. (2018). Lessons learned from analyzing dynamic promotion for user-level threading. In *SC '18: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis.*
- Hammacher, C., Streit, K., Zeller, A., & Hack, S. (2016). Thread-level speculation with kernel support. In *CC 2016: Proceedings of the 25th International Conference on Compiler Construction.*

Discussion (25 points)

Critical Thinking (75 points)

Option 1: Compare and contrast Process Management utilities of Linux OS with a computer's OS.

Compare the Process Management utilities in the Linux OS with that of your main computer's OS. Which one is easier to use and why? Describe the information provided by each and which system provides more useful information. Discuss what you would do differently if you were to create your own Process Management utility.

Your paper should be 2-3 pages and contain 2-3 references. Use the CSU Global library to find additional appropriate references beyond the references given in the modules. Make sure it is in APA Format. A helpful resource for APA formatting can be found at CSU Global Guide to Writing and APA.

Option 2: Compare and contrast Process Management utilities of Linux OS with a mobile phone's OS.

Compare the Process Management utilities in the Linux OS with that of your phone or tablet's OS. Which one is easier to use and why? Describe the information provided by each, and which system provides more useful information. Discuss what you would do differently if you were to create your own Process Management utility?

Your paper should be 2-3 pages and contain 2-3 references, which can include URLs of helpful websites. Use the CSU Global library to find additional appropriate references beyond the references given in the modules. Make sure it is in APA Format. A helpful resource for APA formatting can be found at CSU Global Guide to Writing and APA.

Portfolio Milestone (10 points)

Option #1: Using C Programming language

In your Linux installation, use Bash, or a Linux Shell of your choice, to do the following:

1. Generate a random number: *echo $RANDOM*
2. Output a random number into a file called file1.txt: *echo $RANDOM > file1.txt*
3. Append another random number to the end of this file: *echo $RANDOM >> file1.txt*
4. Remove file1.txt: *rm file1.txt*
5. Create a script called numbers.sh, that does this one thousand (1000) times, using the "for" loop.
6. Make the script executable: *chmod +x numbers.sh*
7. Run the script: *./numbers.sh*

If you are not familiar with Linux Shell scripting, watch the following videos.

McWilliams, G. (2019, April 26) *Introduction to bash* [Video file]. Retrieved from https://www.linkedin.com/learning/linux-system-engineer-bash-shell-scripting-for-automation/introduction-to-bash

Simpson, S. (2013, November 26) *Introducing for loops* [Video file]. Retrieved from https://www.linkedin.com/learning/learning-bash-scripting/introducing-for-loops

You should now have a file called file1.txt containing 1,000 lines, with each line being a random number.

Create a C program to perform this task, to create file2.txt. You should now have 2 files: file1.txt & file2.txt, each containing 1,000 lines, with each line being a random number.

Do a word and line count on these programs, scripts, and text files (feel free to create a new folder(s) to store these, if you prefer a certain level of organization.)

*wc **

Take a screenshot of the files and their word/line counts and submit to the instructor.

Option #2: Using Python Programming language

In your Linux installation, use Bash, or a Linux Shell of your choice, to do the following:

1. Generate a random number: *echo $RANDOM*
2. Output a random number into a file called file1.txt: *echo $RANDOM > file1.txt*
3. Append another random number to the end of this file: *echo $RANDOM >> file1.txt*
4. Remove file1.txt: *rm file1.txt*
5. Create a script called numbers.sh, that does this one thousand (1000) times, using the "for" loop.
6. Make the script executable: *chmod +x numbers.sh*

7. Run the script: *./numbers.sh*

If you are not familiar with Linux Shell scripting, watch the following videos.

McWilliams, G. (2019, April 26) *Introduction to bash* [Video file]. Retrieved from https://www.linkedin.com/learning/linux-system-engineer-bash-shell-scripting-for-automation/introduction-to-bash

Simpson, S. (2013, November 26) *Introducing for loops* [Video file]. Retrieved from https://www.linkedin.com/learning/learning-bash-scripting/introducing-for-loops

You should now have a file called file1.txt containing 1,000 lines, with each line being a random number.

Create a Python program to perform this task, to create file2.txt. You should now have 2 files: file1.txt & file2.txt, each containing 1,000 lines, with each line being a random number.

Do a word and line count on these programs, scripts, and text files (feel free to create a new folder(s) to store these, if you prefer a certain level of organization.)

*wc **

Take a screenshot of the files and their word/line counts and submit to the instructor.

Module 3

Readings

- Chapter 5 in *Operating systems: Internals and design principles*
- Chapter 6 in *Operating systems: Internals and design principles*
- Barbosa, V. C., Diêgo, A., Protti, F., & Souza, U. S. (2016). Deadlock models in distributed computation: Foundations, design, and computational complexity. In *SAC '16: Proceedings of the 31st Annual ACM Symposium on Applied Computing.*
- Morrison, A. (2016). Scaling synchronization in multicore programs. *Communications of the ACM, 59*(11).

Discussion (25 points)

Critical Thinking (75 points)

Option #1: Case Studies of Large Data Files

In the Portfolio Project, you are working with data files containing random numbers. The data files can be anywhere from a few lines (rows) to millions or billions of rows. Creating and processing such files efficiently has very practical applications. For example, instead of random numbers, a company might have names and other information for its employees, which can be in the hundreds of thousands.

Describe, in detail, 3 real-world examples of possible data files of at least one million rows, and why it would be important for a business to process such files in fractions of a second instead of several seconds. You can use arenas such as financial systems, inventory tracking, state drivers license information, or anything else that you can think of. How would you use the methods that were covered in this and the previous module to optimize processing times for such files?

Your paper should be 2-3 pages in length and conform to *CSU Global Guide to Writing and APA*. Include references in addition to the course textbook. The CSU Global Library is a good place to find these references.

Option #2: Case studies of very large data files

In the Portfolio Project, you are working with data files containing random numbers. The data files can be anywhere from a few lines (rows) to millions or billions of rows. Creating and processing such files efficiently has very practical applications. For example, instead of random numbers, a company might have names and other information for its employees, which can be in the hundreds of thousands.

Describe, in detail, 2 real-world examples of possible data files of at least one billion rows, how long you think it takes, on average, to process them, and why it would be important for a business to be able to process such files as quickly as possible. You can use arenas such as social media, banking, worldwide weather statistics, or anything else that you can think of. How would you use the methods that were covered in this and the previous module to optimize processing times for such files?

Your paper should be 2-3 pages in length and conform to *CSU Global Guide to Writing and APA*. Include at least references in addition to the course textbook. The CSU Global Library is a good place to find these references.

Portfolio Milestone (50 points)

Option #1: Using C Programming language

In your Linux installation, modify the numbers.sh script, to do the following:

1. Add to the script commands to display the system time, before and after the process. Optional: store the system time in variables, to display just the difference at the end of the process, i.e. how many hours/minutes/seconds it took to run this script. Hint: use the built-in SECONDS variable to do most of the work.
2. Modify the "for" loop to repeat one million (1,000,000) times.
3. Save the script and delete file1.txt that was created from the previous exercise.
4. Run the script.

You should now have a file called file1.txt containing one million lines, with each line being a random number. You should also have information indicating how long this process took to run.

Create a C program to perform this task, to create file2.txt, and compare execution times. Can you use what you have learned in the last 2 modules, like multithreading or synchronization, to make this process run faster? How would you do that? Use at least 2 other methods to try to improve execution time.

Write down the summary of the results, with the following information:

1. Describe the different methods you used to perform this task, with the times each one took.
2. Explain why you chose each of those methods.
3. Did each of these methods perform as you expected them to, or were there any surprises? Describe your findings in detail.
4. If your system had double its current processing power (CPU power), how much of an improvement would you expect for this process? Explain the reasons for that estimate, and provide references to support your opinion.

Option #2: Using Python Programming language

In your Linux installation, modify the numbers.sh script, to do the following:

1. Add to the script commands to display the system time, before and after the process. Optional: store the system time in variables, to display just the difference at the end of the process, i.e. how many hours/minutes/seconds it took to run this script. Hint: use the built-in SECONDS variable to do most of the work.
2. Modify the “for” loop to repeat one million (1,000,000) times.
3. Save the script and delete file1.txt that was created from the previous exercise.
4. Run the script.

You should now have a file called file1.txt containing one million lines, with each line being a random number. You should also have information indicating how long this process took to run.

Create a Python program to perform this task, to create file2.txt, and compare execution times. Can you use what you've learned in the last 2 modules, like multithreading or synchronization, to make this process run faster? How would you do that? Use at least 2 other methods to try to improve execution time.

Write down the summary of the results, with the following information:

1. Describe the different methods you used to perform this task, with the times each one took.
2. Explain why you chose each of those methods.
3. Did each of these methods perform as you expected them to, or were there any surprises? Describe your findings in detail.
4. If your system had double its current processing power (CPU power), how much of an improvement would you expect for this process? Explain the reasons for that estimate, and provide references to support your opinion.

Module 4

Readings

- Chapter 7 in *Operating systems: Internals and design principles*
- Kültürsay, E., Ebcio, K., Küçük, G., & Kandemir, M. T. (2016). Memory partitioning in the limit. *International Journal of Parallel Programming*, *44*(2), 337–380. (To view this reading, please open the link provided and download the "Full PDF Text.")
- Liu, Y., Kato, S., & Edahiro, M. (2018). Is the heap manager important to many cores? In *ROSS'18: Proceedings of the 8th International Workshop on Runtime and Operating Systems for Supercomputers.*

Discussion (25 points)

Critical Thinking (75 points)

Option #1: Comparing Best-Fit to First-Fit Placement Algorithms

Create a PowerPoint presentation with at least 5 slides that demonstrates why the best-fit algorithm ends up performing worse than the first-fit algorithm in allocating memory to processes. You can use "before" and "after" snapshots of the scenarios that you put forth.

Write a paper explaining how each algorithm works for your scenarios, how many processes can be loaded with each algorithm before requiring memory compaction, and any other determining factors that would influence you in selecting one algorithm over another. Can you think of situations where the best-fit algorithm is preferable to the first-fit algorithm? Explain your reasons for that opinion.

Your paper should be 2-3 pages in length and conform to *CSU Global Guide to Writing and APA*. Include at least four references in addition to the course textbook. The CSU Global Library is a good place to find these references.

Option #2: Comparing Best-Fit to Next-Fit Placement Algorithms

Create a PowerPoint presentation with at least 5 slides, that demonstrates why the best-fit algorithm ends up performing worse than the next-fit algorithm in allocating memory to processes. You can use "before" and "after" snapshots of the scenarios that you put forth.

Write a paper explaining how each algorithm works for your scenarios, how many processes can be loaded with each algorithm before requiring memory compaction, and any other determining factors that would influence you in selecting one algorithm over another. Can you think of situations where the best-fit algorithm is preferable to the next-fit algorithm? Explain your reasons for that opinion.

Your paper should be 2-3 pages in length and conform to *CSU Global Guide to Writing and APA*. Include at least four references in addition to the course textbook. The CSU Global Library is a good place to find these references.

Portfolio Milestone (25 points)

Option #1: Using C Programming language

From the previous exercises, you have 2 files in your Linux installation: file1.txt and file2.txt, both with one million rows of random numbers. Create a new script, double_numbers.sh, to read each line of file1.txt, storing the number from each line into a variable, then write a value that is double the original number, into another file called newfile1.txt. Once this is done for the entire contents of file1.txt, display the time it took to run. Your script might look as follows:

```
#!/bin/bash

SECONDS=0

while read -r number

do

  let "number *= 2"

  echo $number >> newfile1.txt

done < file1.txt

duration=$SECONDS

echo $duration
```

You may want to create a smaller file to work with until the process is working correctly. You can use the "head" command to grab the first 10 rows of file1.txt into another file and use that file in your script:

```
head file1.txt > tempfile.txt
```

For proper syntax of reading files into variables, read *How to read file line by line in Bash script*.

You can also use the "head" command to quickly verify that newfile1.txt contains numbers that are double the value of the corresponding rows of file1.txt.

```
head file1.txt newfile1.txt
```

Once you have the script working to your satisfaction, delete newfile1.txt and run double_numbers.sh. You should now have a file called newfile1.txt containing one million lines. You should also have information indicating how long this process took to run.

Using C, create a program to perform this task, with 3 methods of doing this:

1. Read the entire contents of file1.txt into memory, then process each row.
2. Read one row of file1.txt at a time and process it.
3. Split file1.txt into 2 parts and read each part into memory separately.

Be sure to capture execution times for each method. How do they compare to each other, and to that of the double_numbers.sh script? Were there any surprises for you, or did the results match your expectations? Describe your findings in detail.

Option #2: Using Python Programming language

From the previous exercises, you have 2 files in your Linux installation: file1.txt and file2.txt, both with one million rows of random numbers. Create a new script, double_numbers.sh, to read each line of file1.txt, storing the number from each line into a variable, then write a value that is double the original number into another file called newfile1.txt. Once this is done for the entire contents of file1.txt, display the time it took to run. Your script might look as follows:

```
#!/bin/bash

SECONDS=0

while read -r number

do

  let "number *= 2"

  echo $number >> newfile1.txt

done < file1.txt

duration=$SECONDS

echo $duration
```

You may want to create a smaller file to work with until the process is working correctly. You can use the "head" command to grab the first 10 rows of file1.txt into another file, and use that file in your script.

head file1.txt > tempfile.txt

For proper syntax of reading files into variables, read *How to read file line by line in Bash script*.

You can also use the "head" command to quickly verify that newfile1.txt contains numbers that are double the value of the corresponding rows of file1.txt:

head file1.txt newfile1.txt

Once you have the script working to your satisfaction, delete newfile1.txt and run double_numbers.sh. You should now have a file called newfile1.txt containing one million lines. You should also have information indicating how long this process took to run.

Using Python, create a program to perform this task, with 3 methods of doing this:

1. Read the entire contents of file1.txt into memory, then process each row.
2. Read one row of file1.txt at a time and process it.
3. Split file1.txt into 2 parts and read each part into memory separately.

Be sure to capture execution times for each method. How do they compare to each other, and to that of the double_numbers.sh script? Were there any surprises for you, or did the results match your expectations? Describe your findings in detail.

Module 5

Readings

- Chapter 8 in *Operating systems: Internals and design principles*
- Alam, H., Zhang, T., Erez, M., & Etsion, Y. (2018). Do-it-yourself virtual memory translation. *ACM SIGOPS Operating Systems Review, 50*(1).
- Haria, S., Hill, M. D., Swift, M. M. (2018). Devirtualizing memory in heterogeneous systems. In *ASPLOS '18: Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems.*

Discussion (25 points)

Critical Thinking (75 points)

Option #1: Evaluate and experiment with virtual memory settings on a computer

In your computer's OS, locate the settings for virtual memory. Are they set to the optimal setting? What happens when you increase the size to be double that of the current setting? What happens when you turn off virtual memory? How does the size of your RAM affect the recommendations for optimal virtual memory settings?

Describe how it affects the performance of your computer, to use different virtual memory sizes, or to not use it at all. Which resource-intensive applications are affected, and which ones are not? Be sure to include the identifying information about your OS, such as the hardware it is running on, OS name & version, processor, system type, and the size of the RAM.

Your paper should be 2-3 pages in length and conform to *CSU Global Guide to Writing and APA*. Include at least 2-3 references in addition to the course textbook. The CSU Global Library is a good place to find these references.

Option #2: Evaluate and experiment with virtual memory settings on a mobile phone

For this option, you may want to use an old phone that you are no longer using or are planning to upgrade soon, as you will most likely need to jailbreak it, install special apps, and change other system settings that may shorten the phone's lifespan, in order to modify its virtual memory. What happens when you set the size to be one-half of the phone's RAM? What happens when you set the size to be the same as the phone's RAM? What about twice the size? Can you set it to more than that? Is it advisable to do that for your specific phone?

Describe how each setting affects the loading and running of apps. Can you run more, or fewer, resource-intensive apps at the same time, than with the original setting? What else do you notice about your phone's performance, i.e. restart times, battery life, etc.? Be sure to include the identifying information about your phone, such as the hardware it is running on, OS name & version, and the size of the RAM.

Your paper should be 2-3 pages in length and conform to *CSU Global Guide to Writing and APA*. Include at least 2-3 references in addition to the course textbook. The CSU Global Library is a good place to find these references.

Module 6

Readings

- Chapter 9 in *Operating systems: Internals and design principles*
- Chapter 10 in *Operating systems: Internals and design principles*
- Baptiste Lepers, B., Zwaenepoel, W. E., Lozi, J-P., Palix, N., Gouicem, R., Sopena, J., Lawall, J. L., & Muller, G. (2017). Towards proving optimistic multicore schedulers. In *HotOS '17: Proceedings of the 16th Workshop on Hot Topics in Operating Systems.*
- Dinda, P., Wang, X., Wang, J., Beauchene, C., & Hetland, C. (2018). Hard real-time scheduling for parallel run-time systems. In *HPDC '18: Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing.*

Discussion (25 points)

Critical Thinking (75 points)

Option #1: Evaluate real-time process scheduling

In an earlier module, you created programs that read the contents of a large file and process it, writing the results into another large file. What if the files were 10x bigger, i.e. instead of a million rows, they were 10 million rows? Which of the following methods would have the fastest processing time:

1. Run the process as it is, with the larger files.
2. Break the input file up into 10 files and schedule the process on each one to run in real-time, then combine the resulting files into a single output file.

3. Break the input file up into 2 files and schedule the process on each one to run in real-time, then combine the resulting files into a single output file.
4. Break the input file up into 5 files and schedule the process on each one to run in real-time, then combine the resulting files into a single output file.
5. Break the input file up into 20 files and schedule the process on each one to run in real-time, then combine the resulting files into a single output file.

Can you think of other ways to increase efficiency and reduce processing time? Describe in detail or provide a script to do so, with expected results for each method. Feel free to create such scripts and run them, to have actual results instead of theoretical results.

Your paper should be 2-3 pages in length and conform to *CSU Global Guide to Writing and APA*. Include at least 2-3 references in addition to the course textbook. The CSU Global Library is a good place to find these references.

Option #2: Evaluate staggered process scheduling

In an earlier module, you created programs that read the contents of a large file and process it, writing the results into another large file. What if the files were 10x bigger, i.e. instead of a million rows, they were 10 million rows? Which of the following methods would have the fastest processing time:

1. Run the process as it is, with the larger files.
2. Break the files up into 10 files and schedule processes to run 30 seconds or 1 minute apart, then combine the resulting files into a single output file.
3. Break the files up into 2 files and schedule processes to run 30 seconds or 1 minute apart, then combine the resulting files into a single output file.
4. Break the files up into 5 files and schedule processes to run 30 seconds or 1 minute apart, then combine the resulting files into a single output file.
5. Break the files up into 20 files and schedule processes to run 30 seconds or 1 minute apart, then combine the resulting files into a single output file.

Can you think of other ways to increase efficiency and reduce processing time? Describe in detail or provide a script to do so, with expected results for each method. Feel free to create such scripts and run them, to have actual results instead of theoretical results.

Your paper should be 2-3 pages in length and conform to *CSU Global Guide to Writing and APA*. Include at least 2-3 references in addition to the course textbook. The CSU Global Library is a good place to find these references.

Module 7

<u>Readings</u>

- Chapter 11 in *Operating systems: Internals and design principles*
- Akar, N., Tunç, Ç., Gaertner, M., & Erden, F. (2017). Disk scheduling with shortest cumulative access time first algorithms. *Turkish Journal of Electrical Engineering & Computer Sciences, 25*(4).
- Behzad, B., Byna, S., Byna, P., & S, M. (2019). Optimizing I/O performance of HPC applications with autotuning. *ACM Transactions on Parallel Computing, 5*(4).

Discussion (25 points)

Module 8

<u>Readings</u>

- Chapter 12 in *Operating systems: Internals and design principles*
- Khusro, S. & Mashwani, S. R. (2016). The life of a file: From cradle to grave or eternity. In *INFOS '16: Proceedings of the 10th International Conference on Informatics and Systems.*
- Petrov, A. (2018). Algorithms behind modern storage systems. *Communications of the ACM, 61*(8).

<u>Discussion (25 points)</u>

<u>Portfolio Project (240 points)</u>

Option #1: Working with Big Data using Multithreading

The goal of this project is to use the concepts taught in this course to develop an efficient way of working with Big Data.

You should have 2 files in your Linux system: *hugefile1.txt* and *hugefile2.txt*, with one billion lines in each one. If you do not, please go back to the Module 7 Portfolio Reminder and complete the steps there.

Create a program, using a programming language of your choice, to produce a new file: *totalfile.txt*, by taking the numbers from each line of the two files and adding them. So, each line in file #3 is the sum of the corresponding line in *hugefile1.txt* and *hugefile2.txt*.

For example, if the first 5 lines of your files look as follows:

```
$ head -5 hugefile*txt

==> hugefile1.txt <==

4131

29929

6483

7659

25003

==> hugefile1.txt <==

8866

19171

11029

4889

27069
```

then the first 5 lines of *totalfile.txt* look like this:

```
$ head -5 totalfile.txt

12997

49100

17512

12548

52072
```

Because the files of such large sizes cannot be read into memory in their entirety at the same time, you need to use concurrency. Reading the files one line at a time will take a long time, so use what you have learned in this course to optimize this process. Be sure to record the amount of time it takes for each version of your program to complete this task.

Optimize the program by using threads, so that you benefit from multiple cores in your CPU. Create a multithreaded program, where each thread works on the next chunk of the file.

Now, break up *hugefile1.txt* and *hugefile2.txt* into 10 files each, and run your process on all 10 sets in parallel. How do the run times compare to the original process?

Explain your methods and results in detail. What conclusions can you make about the different methods of optimizing large file processing? How has the information that you learned in this course helped you to accomplish this task?

Your paper should be 2-3 pages in length and conform to *CSU Global Guide to Writing and APA*. Include 4-6 references in addition to the course textbook. The CSU Global Library is a good place to find these references.

Option #2: Working with Big Data using Parallel Processing

The goal of this project is to use the concepts taught in this course to develop an efficient way of working with Big Data.

You should have 2 files in your Linux system: *hugefile1.txt* and *hugefile2.txt*, with one billion lines in each one. If you do not, please go back to the Module 7 Portfolio Reminder and complete the steps there.

Create a program, using a programming language of your choice, to produce a new file: *totalfile.txt*, by taking the numbers from each line of the two files and adding them. So, each line in file #3 is the sum of the corresponding line in *hugefile1.txt* and *hugefile2.txt*.

For example, if the first 5 lines of your files look as follows:

```
$ head -5 hugefile*txt

==> hugefile1.txt <==

4131

29929
```

```
6483

7659

25003

==> hugefile1.txt <==

8866

19171

11029

4889

27069
```

then the first 5 lines of *totalfile.txt* look like this:

```
$ head -5 totalfile.txt

12997

49100

17512

12548

52072
```

Because the files of such large sizes cannot be read into memory in their entirety at the same time, you need to use concurrency. Reading the files one line at a time will take a long time, so use what you learned in this course to optimize this process. Be sure to record the amount of time it takes for each version of your program to complete this task.

Create two programs, where one program reads the first half of the files, and another program reads the second half. Use the OS to launch both programs simultaneously.

Now, break up *hugefile1.txt* and *hugefile2.txt* into 10 files each, and run your process on all 10 sets in parallel. How do the run times compare to the original process?

Explain your methods and results in detail. What conclusions can you make about the different methods of optimizing large file processing? How has the information that you learned in this course helped you to accomplish this task?

Your paper should be 2-3 pages in length and conform to *CSU Global Guide to Writing and APA*. Include 4-6 references in addition to the course textbook. The CSU Global Library is a good place to find these references.

Course Policies

Grading Scale	
A	95.0 – 100
A-	90.0 – 94.9
B+	86.7 – 89.9
B	83.3 – 86.6
B-	80.0 – 83.2
C+	75.0 – 79.9
C	70.0 – 74.9
D	60.0 – 69.9
F	59.9 or below

Course Grading

20% Discussion Participation
45% Critical Thinking Assignments
35% Final Portfolio Project

In-Classroom Policies

For information on late work and incomplete grade policies, please refer to our **In-Classroom Student Policies and Guidelines** or the Academic Catalog for comprehensive documentation of CSU-Global institutional policies.

Academic Integrity
Students must assume responsibility for maintaining honesty in all work submitted for credit and in any other work designated by the instructor of the course. Academic dishonesty includes cheating, fabrication, facilitating academic dishonesty, plagiarism, reusing /re-purposing your own work (see *CSU-Global Guide to Writing and APA Requirements* for percentage of repurposed work that can be used in an assignment), unauthorized possession of academic materials, and unauthorized collaboration. The CSU-Global Library provides information on how students can avoid plagiarism by understanding what it is and how to use the Library and Internet resources.

Citing Sources with APA Style
All students are expected to follow the *CSU-Global Guide to Writing and APA Requirements* when citing in APA (based on the APA Style Manual, 6th edition) for all assignments. For details on CSU-Global APA style, please review the APA resources within the CSU-Global Library under the "APA Guide & Resources" link. A link to this document should also be provided within most assignment descriptions in your course.

Disability Services Statement
CSU–Global is committed to providing reasonable accommodations for all persons with disabilities. Any student with a documented disability requesting academic accommodations should contact the Disability Resource Coordinator at 720-279-0650 and/or email ada@CSUGlobal.edu for additional information to coordinate reasonable accommodations for students with documented disabilities.

Netiquette
Respect the diversity of opinions among the instructor and classmates and engage with them in a courteous, respectful, and professional manner. All posts and classroom communication must be conducted in accordance with the student code of conduct. Think before you push the Send button. Did you say just what you meant? How will the person on the other end read the words?

Maintain an environment free of harassment, stalking, threats, abuse, insults or humiliation toward the instructor and classmates. This includes, but is not limited to, demeaning written or oral comments of an ethnic, religious, age, disability, sexist (or sexual orientation), or racist nature; and the unwanted sexual advances or intimidations by email, or on discussion boards and other postings within or connected to the online classroom. If you have concerns about something that has been said, please let your instructor know.