CSC139 Operating System Principles

Fall 2021, Part 0

Instructor: Dr. Yuan Cheng

Session Plan

- Course Logistics

Instructor

- Yuan Cheng
  - Office: RVR 5042
  - Email: yuan.cheng@csus.edu
  - Web: http://www.ycheng.org
  - Office hours: Mon 2 pm – 3 pm, Fri 2 pm – 4 pm, or by appointment
  - Research interests: access control; authentication; security & privacy in emerging technologies; usable security

Description

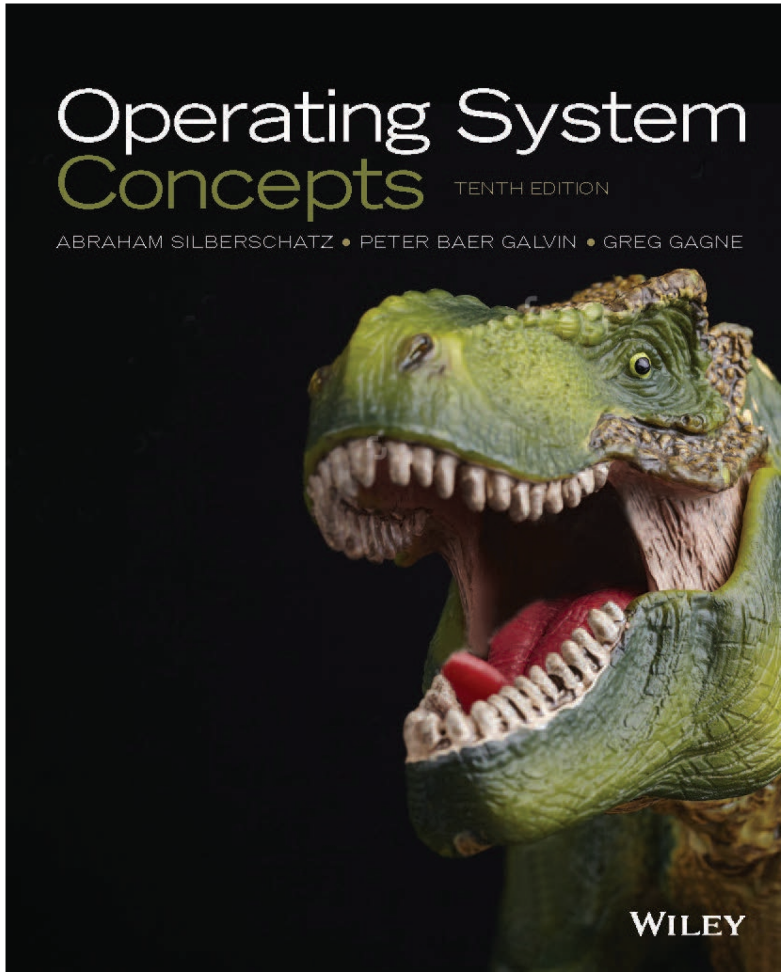
- 3 units
- Lectures from Aug. 31st to Dec. 9th
  - Section 4: TR 12 pm – 1:15 pm
  - Section 5: TR 1:30 pm – 2:45 pm
  - Section 6: TR 4 pm – 5:15 pm
- We will discuss contemporary operating system organization and structure. Topics include: process and thread, concurrency, scheduling, inter-process communication and synchronization, deadlock, real and virtual memory management, device management, file systems, security and protection.

Prerequisites

- CSC 60, CSC 130 and either CSC 137 or CpE 185.
- Be comfortable with C programming language.

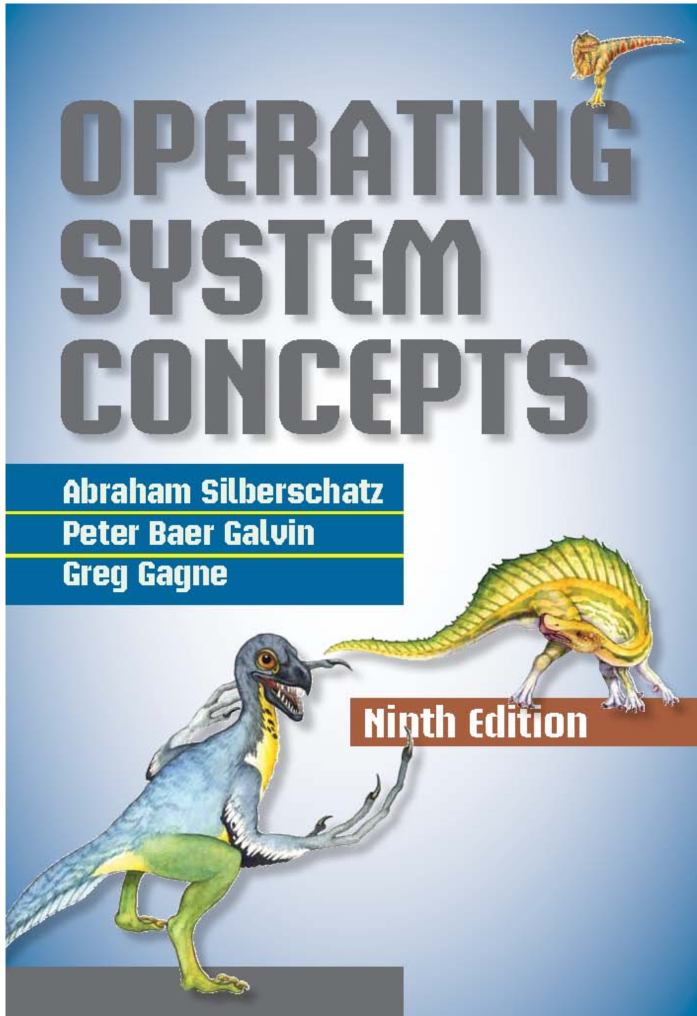
Textbook

- A. Silberschatz, P. Galvin, and G. Gagne, Operating System Concepts, 10th edition, Wiley, 2018. ISBN: 978-1119320913


Operating System Concepts
TENTH EDITION
ABRAHAM SILBERSCHATZ • PETER BAER GALVIN • GREG GAGNE
WILEY

Alternative Textbook

- A. Silberschatz, G. Gagne and P. Galvin, Operating System Concepts, 9th edition, Wiley, 2013. ISBN: 978-1118129388



Resources

- Canvas (for announcements, slides, assignments, and grades, etc.)
- Zoom lectures:
  - Section 4: https://csus.zoom.us/j/84680772901
  - Section 5: https://csus.zoom.us/j/89556327967
  - Section 6: https://csus.zoom.us/j/88501880601
  - Office hours: https://csus.zoom.us/j/9756375621
- Slack channel (for discussions and Q&A):
  - https://join.slack.com/t/csuscsc139fall2021/shared_invite/zt-v6sajrrl-8YeZBB89VjVnAxp7qjQeNA
- Book site: http://os-book.com/OS10/index.html

Course Format

- Online synchronous
  - Recorded lectures
  - Live review, clarification (when necessary), and group worksheets (usually once per chapter)
- (Review) + lecture + (worksheets)
  - A short overview of the previous lecture to make sure the old content is not completely forgotten
  - Worksheet practices to make sure the lecture is well understood

Course Outcomes

- Students completing this course will be able to
  1. Explain the principles of concurrency and the trade-offs in synchronization approaches, and apply different synchronization approaches to the critical section problem and to process coordination.
  2. Articulate different methods for handling deadlocks and starvations.
  3. Grasp issues, principles, performance criteria, and pros/cons of various operating system algorithms for managing different types of system resources.
  4. Write multi-process and multi-thread programs to solve concurrency control and synchronization problems using various types of system calls, API calls, semaphores, mutex locks, condition variables, and IPC methods in Unix/Linux environments.
  5. Identify the security issues in operating systems and how to design around them.
  6. Fully explain how a process and its memory are managed by an operating system.
  7. Demonstrate knowledge of virtualization of computing hardware.
  8. Give detailed information about file systems in modern operating systems.

Tentative Topics

- Processes and threads.
- Race condition, the critical section problem and its solutions.
- Deadlock and starvation.
- Processor scheduling and real-time scheduling.
- Memory management and virtual memory.
- File system, disk scheduling and RAID.
- I/O systems.
- Network and distributed operating systems.
- Protection and security.

Grading

- Midterms: 30% (15% × 2)
- Final Exam: 20%
- Programming Assignments: 28%
- Homework Assignments: 10%
- Quizzes: 4%
- Participation: 8%

Letter Grade Assignment

- The final letter grade will be *roughly* based on the final **percentage** as follows:

Letter Grade	**Percentage**	**Performance**
A	93-100%	Excellent Work
A-	90-92%	Nearly Excellent Work
B+	87-89%	Very Good Work
B	83-86%	Good Work
B-	80-82%	Mostly Good Work
C+	77-79%	Above Average Work
C	73-76%	Average Work
C-	70-72%	Mostly Average Work
D	67-69%	Below Average Work
D-	60-66%	Poor Work
F	0-59%	Failing Work

Historical Data

Section	Median	Average	Highest	Stdev	D/F/W Rate
F18 Sec 3	66.33	66.33	92.69	17.21	7/38
F18 Sec 4	67.58	67.59	86.88	12.55	3/39
S19 Sec 2	75.40	73.19	85.40	8.90	2/32
S19 Sec 4	74.92	75.51	90.06	8.55	4/39
S19 Sec 6	74.80	72.55	94.02	14.20	5/38
S19 Sec 7	73.25	73.53	89.68	9.16	5/36
F19 Sec 1	69.34	68.16	94.39	12.78	5/37
S20 Sec 2	70.09	68.15	81.78	9.82	8/37
S20 Sec 3	73.26	71.48	86.14	9.28	5/35
F20 Sec 1	68.47	64.51	85.33	16.28	8/36
F20 Sec 2	69.86	64.77	93.32	17.97	8/34
F20 Sec 3	65.68	67.80	95.3	10.20	6/34

Percentile ≈ Curve

Letter Grade	Percentage	Actual cutoffs in a past section
A	93-100%	81-100% (4)
A-	90-92%	76-81% (4)
B+	87-89%	71-76% (3)
B	83-86%	70-71% (3)
B-	80-82%	66-71% (3)
C+	77-79%	65-66% (2)
C	73-76%	62-65% (5)
C-	70-72%	60-62% (4)
D	60-69%	55-60% (4)
F	0-59%	0-55% (2)

Attendance

- Attendance to class is expected. Class roll will not be checked after first two weeks of classes. However, you are responsible for material presented and announcements made in class or by email. This could include changes to the syllabus, exam dates, etc.

- You will not be penalized for absences unless you are unable to complete course learning outcomes. You will not be asked to provide the instructor formal documentation from a healthcare provider.

Assignments

- Complete Assignments
  - Work by yourself
  - Electronic submission
  - Submit on time (All assignments due at 11:59pm on the specified due date)
  - ***15% penalty per day***
  - ***No acceptance if more than 72 hours late***
  - ***5 grace days to spare***
- If COVID related illness/exposure results in any missed course work, you should *proactively work with the instructor to plan make-up work*. It remains your responsibility to complete any missed work. But please realize that class will continue, and you may find yourself in the situation where you are unable to complete all work by the end of the semester. In such a case, you should consider a late withdrawal or an incomplete grade.

Exams

- Midterm and Final Exam
  - Online
  - No webcam enforced
  - I am still trying to figure out a better plan…
- Make-up Exams
  - Make-up exams will only be given under extreme circumstances. The instructor *reserves the right* to reject make-up requests. There will be *no* make-up for unannounced quizzes under any circumstances.

Drop and Incomplete Policies

- Drop policy
  - Students wishing to withdraw from all courses should fill out the Semester Withdrawal Form.
  - Until the end of the second week of instruction of the semester, students are expected to drop courses by using "My Sac State" (http://www.my.csus.edu). Students will be charged registration fees for all courses not dropped prior to the first day of instruction. The drop in units refund deadline is the end of the second week of instruction.
  - Drops during the third and fourth weeks of instruction are processed in the academic department offering the course and require instructor and department chair approval. Forms are available at academic department offices, or at the Office of the Registrar's website (https://www.csus.edu/student-affairs/centers-programs/student-services-center/forms.html).
- Incomplete policy
  - Under emergency/special circumstances, students may petition for an incomplete grade. It is the responsibility of the student to bring pertinent information to the attention of the instructor and to determine from the instructor the remaining course requirements that must be satisfied to remove the Incomplete. A final grade is assigned when the work agreed upon has been completed and evaluated. All incomplete course assignments must be completed within 12 months.

Academic Honesty

- “The principles of truth and honesty are recognized as fundamental to a community of scholars and teachers. California State University, Sacramento expects that both faculty and students will honor these principles, and in so doing, will protect the integrity of academic work and student grades.”
- Read more about Sac State's Academic Honesty Policy & Procedures
- **Definitions**
  - At Sac State, “**cheating** is the act of obtaining or attempting to obtain credit for academic work through the use of any dishonest, deceptive, or fraudulent means.”
  - **Plagiarism** is a form of cheating. At Sac State, “plagiarism is the use of distinctive ideas or works belonging to another person without providing adequate acknowledgement of that person’s contribution.”

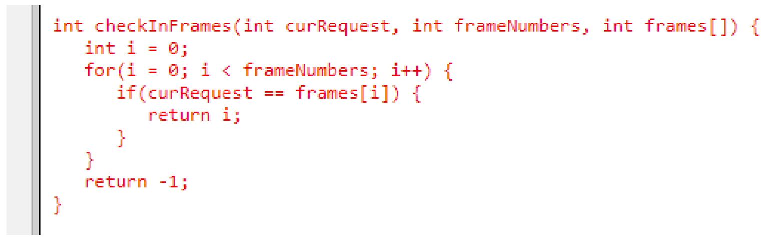
Academic Honesty (cont.)

- Plagiarism at Sacramento State includes but is not limited to:
  1. The act of incorporating into one’s own work the ideas, words, sentences, paragraphs, or parts thereof, or the specific substance of another’s work without giving appropriate credit thereby representing the product as entirely one's own. Examples include not only word-for-word copying, but also the "mosaic" (i.e., interspersing a few of one’s own words while, in essence, copying another’s work), the paraphrase (i.e., rewriting another’s work while still using the other’s fundamental idea or theory); fabrication (i.e., inventing or counterfeiting sources), ghost-writing (i.e., submitting another’s work as one’s own) and failure to include quotation marks on material that is otherwise acknowledged; and
  2. Representing as one’s own another’s artistic or scholarly works such as musical compositions, computer programs, photographs, paintings, drawing, sculptures, or similar works.

Code Plagiarism Examples (1)

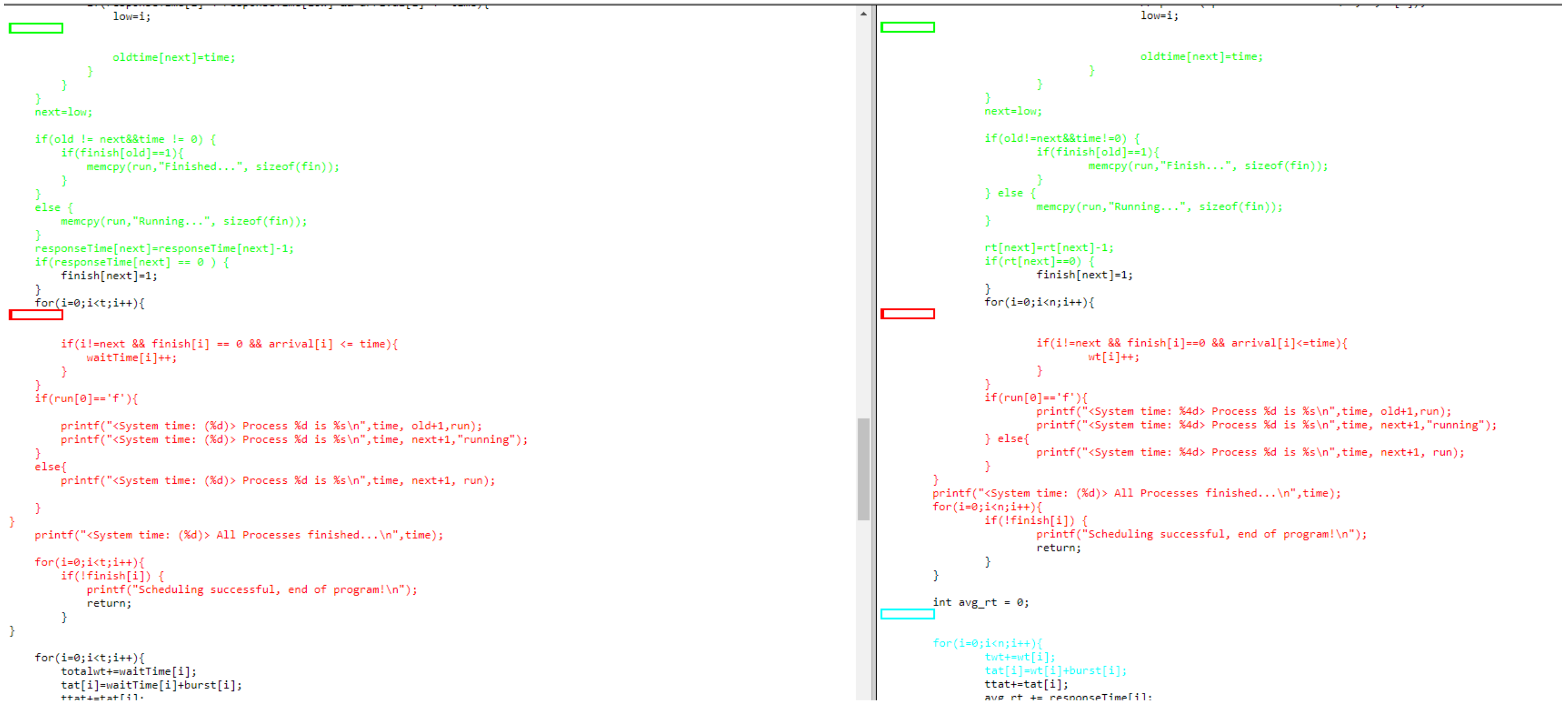
```
int existsAlreadyOPT(int requestArrayValOPT, int numOfFrames ,int frameArrayOPT[]){
        int j = 0;
        for(j = 0; j<numOfFrames; j++) {
                if(requestArrayValOPT == frameArrayOPT[j])

                return j;
        }
        return -1;
}
```

```
// helper functions
```



```
int checkInFrames(int curRequest, int frameNumbers, int frames[]) {
   int i = 0;
   for(i = 0; i < frameNumbers; i++) {
      if(curRequest == frames[i]) {
         return i;
      }
   }
   return -1;
}
```

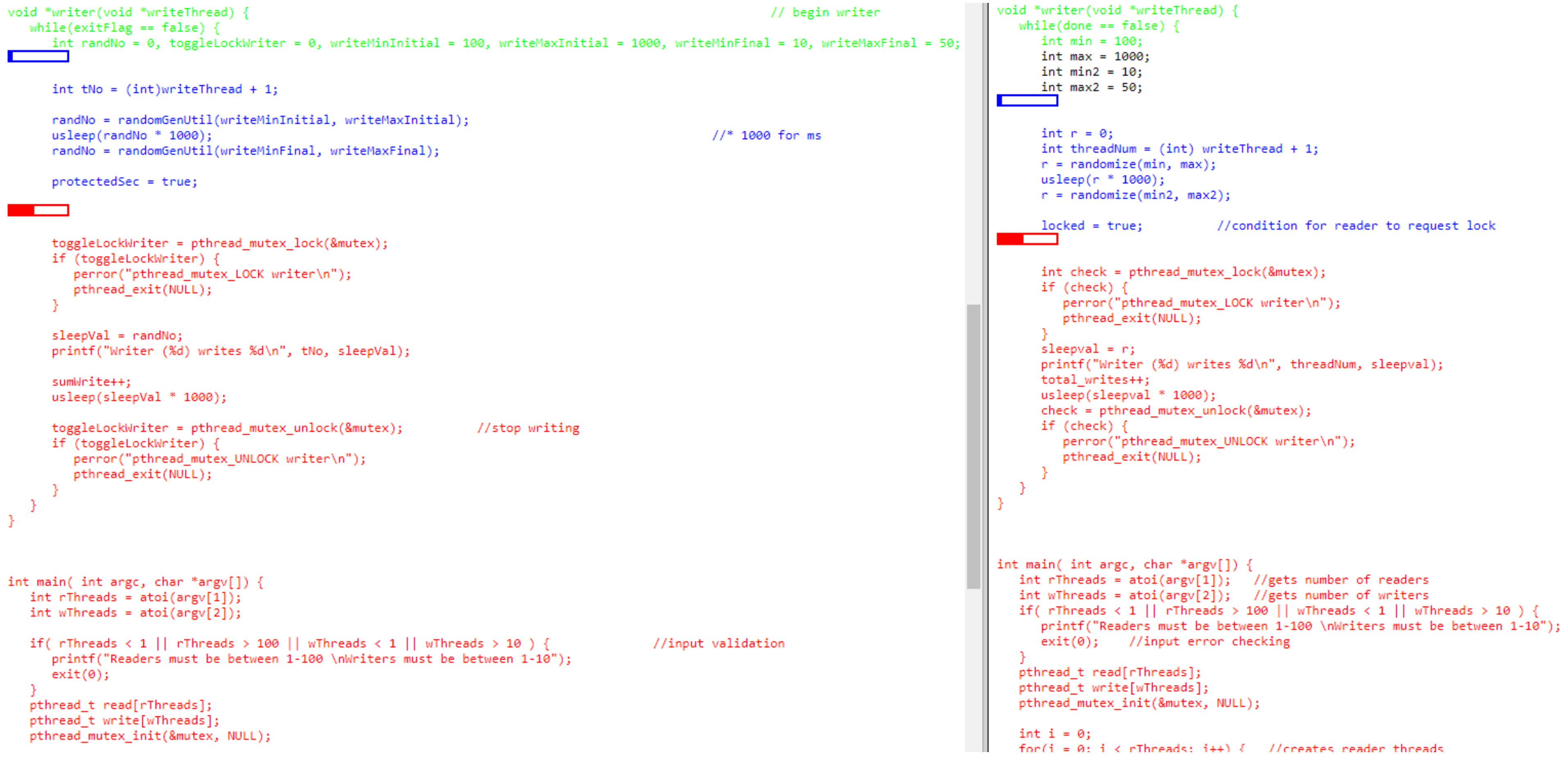
Code Plagiarism Examples (2)



```
                low=i;


                oldtime[next]=time;
            }
        }
    }
    next=low;

    if(old != next&&time != 0) {
        if(finish[old]==1){
            memcpy(run,"Finished...", sizeof(fin));
        }
    }
    else {
        memcpy(run,"Running...", sizeof(fin));
    }
    responseTime[next]=responseTime[next]-1;
    if(responseTime[next] == 0 ) {
        finish[next]=1;
    }
    for(i=0;i<t;i++){


        if(i!=next && finish[i] == 0 && arrival[i] <= time){
            waitTime[i]++;
        }
    }
    if(run[0]=='f'){

        printf("<System time: (%d)> Process %d is %s\n",time, old+1,run);
        printf("<System time: (%d)> Process %d is %s\n",time, next+1,"running");
    }
    else{
        printf("<System time: (%d)> Process %d is %s\n",time, next+1, run);

    }
}
    printf("<System time: (%d)> All Processes finished...\n",time);

    for(i=0;i<t;i++){
        if(!finish[i]) {
            printf("Scheduling successful, end of program!\n");
            return;
        }
}

    for(i=0;i<t;i++){
        totalwt+=waitTime[i];
        tat[i]=waitTime[i]+burst[i];
        ttat+=tat[i];
```

```
                                low=i;


                                oldtime[next]=time;
                        }
                }
        }
        next=low;

        if(old!=next&&time!=0) {
                if(finish[old]==1){
                        memcpy(run,"Finish...", sizeof(fin));
                }
        } else {
                memcpy(run,"Running...", sizeof(fin));
        }

        rt[next]=rt[next]-1;
        if(rt[next]==0) {
                finish[next]=1;
        }
        for(i=0;i<n;i++){


                if(i!=next && finish[i]==0 && arrival[i]<=time){
                        wt[i]++;
                }
        }
        if(run[0]=='f'){
                printf("<System time: %4d> Process %d is %s\n",time, old+1,run);
                printf("<System time: %4d> Process %d is %s\n",time, next+1,"running");
        } else{
                printf("<System time: %4d> Process %d is %s\n",time, next+1, run);
        }
}
printf("<System time: (%d)> All Processes finished...\n",time);
for(i=0;i<n;i++){
        if(!finish[i]) {
                printf("Scheduling successful, end of program!\n");
                return;
        }
}

int avg_rt = 0;


for(i=0;i<n;i++){
        twt+=wt[i];
        tat[i]=wt[i]+burst[i];
        ttat+=tat[i];
        avg_rt += responseTime[i];
```

Code Plagiarism Examples (3)



```
void *writer(void *writeThread) {                                                        // begin writer
   while(exitFlag == false) {
      int randNo = 0, toggleLockWriter = 0, writeMinInitial = 100, writeMaxInitial = 1000, writeMinFinal = 10, writeMaxFinal = 50;


      int tNo = (int)writeThread + 1;

      randNo = randomGenUtil(writeMinInitial, writeMaxInitial);
      usleep(randNo * 1000);                                                        //* 1000 for ms
      randNo = randomGenUtil(writeMinFinal, writeMaxFinal);

      protectedSec = true;


      toggleLockWriter = pthread_mutex_lock(&mutex);
      if (toggleLockWriter) {
         perror("pthread_mutex_LOCK writer\n");
         pthread_exit(NULL);
      }

      sleepVal = randNo;
      printf("Writer (%d) writes %d\n", tNo, sleepVal);

      sumWrite++;
      usleep(sleepVal * 1000);

      toggleLockWriter = pthread_mutex_unlock(&mutex);          //stop writing
      if (toggleLockWriter) {
         perror("pthread_mutex_UNLOCK writer\n");
         pthread_exit(NULL);
      }
   }
}



int main( int argc, char *argv[]) {
   int rThreads = atoi(argv[1]);
   int wThreads = atoi(argv[2]);

   if( rThreads < 1 || rThreads > 100 || wThreads < 1 || wThreads > 10 ) {              //input validation
      printf("Readers must be between 1-100 \nWriters must be between 1-10");
      exit(0);
   }
   pthread_t read[rThreads];
   pthread_t write[wThreads];
   pthread_mutex_init(&mutex, NULL);
```

```
void *writer(void *writeThread) {
   while(done == false) {
      int min = 100;
      int max = 1000;
      int min2 = 10;
      int max2 = 50;


      int r = 0;
      int threadNum = (int) writeThread + 1;
      r = randomize(min, max);
      usleep(r * 1000);
      r = randomize(min2, max2);

      locked = true;          //condition for reader to request lock


      int check = pthread_mutex_lock(&mutex);
      if (check) {
         perror("pthread_mutex_LOCK writer\n");
         pthread_exit(NULL);
      }
      sleepval = r;
      printf("Writer (%d) writes %d\n", threadNum, sleepval);
      total_writes++;
      usleep(sleepval * 1000);
      check = pthread_mutex_unlock(&mutex);
      if (check) {
         perror("pthread_mutex_UNLOCK writer\n");
         pthread_exit(NULL);
      }
   }
}



int main( int argc, char *argv[]) {
   int rThreads = atoi(argv[1]);   //gets number of readers
   int wThreads = atoi(argv[2]);   //gets number of writers
   if( rThreads < 1 || rThreads > 100 || wThreads < 1 || wThreads > 10 ) {
      printf("Readers must be between 1-100 \nWriters must be between 1-10");
      exit(0);    //input error checking
   }
   pthread_t read[rThreads];
   pthread_t write[wThreads];
   pthread_mutex_init(&mutex, NULL);

   int i = 0;
   for(i = 0; i < rThreads; i++) {   //creates reader threads
```

Accommodations

- If you have a documented disability and verification from the Office of Services to Students with Disabilities (SSWD), and wish to discuss academic accommodations, please contact your instructor as soon as possible. It is the student's responsibility to provide documentation of disability to SSWD and meet with a SSWD counselor to request special accommodation *before* classes start.
- SSWD is located in Lassen Hall 1008 and can be contacted by phone at (916) 278-6955 (Voice) (916) 278-7239 (TDD only) or via email at sswd@csus.edu.
- Campus policy regarding religious observances requires that faculty make every effort to reasonably and fairly deal with all students who, because of religious obligations, have conflicts with scheduled exams, assignments or required attendance. In case of conflict with a test, please let me know at least two weeks in advance.

Basic Needs Support

- If you are experiencing challenges with food, housing, financial, or other unique circumstances that are impacting your education, help is just a phone call or email away! The CARES office provides case management support for any enrolled student. Email the CARES office at cares@csus.edu to speak with a case manager about the resources available to you. Check out the CARES website.

Health and Safety Information

- If you are sick, stay home. Notify your instructor. Please self-diagnose if you are experiencing any COVID-like symptoms (fever, cough, sore throat, muscle aches, loss of smell or taste, nausea, diarrhea, or headache) or have had exposure to someone who has tested positive for COVID. Contact Student Health & Counseling Services (SHCS) at (916) 278-6461 to receive guidance and/or medical care. The CDC provides a good source of information regarding COVID-19 and a way to self-check symptoms: https://www.cdc.gov/coronavirus/2019-ncov/index.html

Zoom Netiquette

- Be punctual
- Test all technology (e.g., microphone, speaker/headset, Wi-Fi) before the class.
- Mute your audio if you are not speaking.
- “Raise hand” and wait to be called upon.
- Don’t interrupt other people when they’re speaking (or attempt to speak over them).
- Although live access to your webcam is not required in this class, present a professional image as if video is used.
- If you need to share your screen, make sure to close any windows not related to the class. Avoid “screen shuffling” and any potentially embarrassing moments if possible.
- Keep the chatroom appropriate.

Hornet Learning Online 101

- Enroll in [Hornet Learning Online 101]
  - A short, interactive, 1-hour, self-enrolled tutorial
  - Content includes: Growth mindset, time management, technology readiness, Canvas navigation, and an interactive list of campus resources
  - Online readiness survey report and a “digital badge”
- Completing the tutorial and submitting a copy of the badge will give you 2-point extra credit!

Expectations

- Attend class
  - Ask questions
  - Speak up in class
  - Engage during in-class exercises
- Read assigned text
  - Skim readings before lecture
  - Read in-depth later
  - Class/slides/notes are for high-level concepts
  - Reading & exercises are for low-level details
  - Exams rely on low-level details
- Start assignments early
  - “If you wait until the last minute, it takes only a minute to do.” -- Cyril Northcote Parkinson
- Write well and clearly
- Get help when you need it

How to submit a programming assignment?

Make sure your code can be compiled and work on `ecs-pa-coding1.ecs.csus.edu`

- You may need to connect to the campus VPN before accessing the server from off campus

Upload to Canvas the following:

- All the files necessary to compile, link, and run your program.
- A document describing how to run the program you created. Call this document `README.TXT`.
- These files should be placed in a directory called `<SacLink username>-asgmt1`.
- Use tar command to place all the files in a single file called `<SacLink username>-asgmt1.tar`. Assuming you are in the directory do the following:`<SacLink username>-asgmt1`
    - Goto the parent directory: `cd ..`
    - tar the files: `tar -cvf <SacLink username>-asgmt1.tar ./<SacLink username>-asgmt1`
    - Verify the files have been placed in a tar file: `tar -tvf <SacLink username>-asgmt1.tar`
    - Compress the files using gzip: `gzip <SacLink username>-asgmt1.tar`
    - Verify that the gzipped file exists: `ls <SacLink username>-asgmt1.tar.gz`

Note: `<SacLink username>` is just a placeholder. Replace it with your SacLink username, e.g., `johndoe`.

Next session

- We will discuss:
  - Introduction to Operating Systems
- Reading assignment: (skim through before class and continue reading over the weekend)
  - SGG: Ch. 1, Ch. 2
    - You may skip 1.11, 2.7, 2.9, and 2.10