Dhirubhai Ambani Institute of Information and Communication Technology
Gandhinagar

Systems Software (IT215)

Date: July 03, 2020 Duration: 1 hour Max Marks 35

Read the instructions carefully:

- All questions are compulsory.
- **Every page/sheet** used by you should clearly indicate the following.
  - your roll number and name on top left
  - signature in top center
  - page number on top right.
- Any page/sheet with the above information (roll no., name, signature, page number) missing will not be considered for evaluation.
- Answers to all questions should be in the same order as specified in the question paper, to ensure that your answers get suitably evaluated.
- There will be an additional 40 minutes time available after the completion of 1 hour for you to scan and upload your answers as a single PDF file at an appropriate place / site.
- You need to check the readability of your PDF file before uploading it. Not only the answers, your roll number, name etc. information should be readable in your submitted PDF.
- Specifying comments in code snippet is mandatory.

---

1. **Answer the following as directed.** [1X20 = 20]
   (a) If p(t1) and p(t2) are threads of process p, and q(t1) and q(t2) are threads of process q, then the time to switch from either p(t1) to p(t2), or q(t1) to q(t2), or p(t1) to q(t2), or p(t2) to q(t1) is constant. Comment (3 to 4 sentences) on the correctness of this statement.
   (b) Demonstrate using a suitable code snippet about how specific signals could be ignored for your program execution. Also, show how your programs will no more be ignoring those same signals in your same program at a later point of execution.
   (c) Demonstrate the working of sem_post() and sem_wait() calls using suitable pseudocode and flowcharts.
   (d) Demonstrate the conversion of host-to-host packet delivery service into a process-to-process communication using a suitable diagram.
   (e) fork() call allows a programmer to create infinite number of processes. Comment (3 to 4 sentences) on the correctness of this statement with supporting code snippet.
   (f) Establish the significance of using WIF macros using a suitable code snippet. Give suitable comments for the example code clearly indicating your reasons to use WIF macros.
   (g) Use of threads is preferred in implementation of web servers. Comment (3 to 4 sentences) on the correctness of this statement with supporting code snippet.
   (h) It will be difficult to implement datagram sockets without suitable port binding. Comment (3 to 4 sentences) on the correctness of this statement with supporting code snippet.
   (i) Switching across processes created using a fork() system call is kernel dependent. Comment (3 to 4 sentences) on the correctness of this statement with suitable example.
   (j) It is difficult to create peer threads as against peer processes. Comment (3 to 4 sentences) on the correctness of this statement with supporting code snippet.

(k) Construct process graph for the following code snippet.

```
for (int i=0; i<=4, i++) {
   fork();
   printf("%d", i);
}
```

Count the total number of processes and threads based-on your process graph when the above code is executed.

(l) Comment (3 to 4 sentences) on the race conditions that may / may not occur if the following code is executed.

```
void *thrd_fn(void *vargp) {
int id;
id = *((int *)vargp);
printf("Thread %d\n", id);
}
int main() {
pthread_t tid;
int i=1;
pthread_create(&tid, NULL, thrd_fn, &i);
}
```

(m) Comment on the execution of following two pthread_create() calls with suitable reasons. You should clearly reason about the similarities (if any) / differences (if any) in these two calls, and why and when one should be preferred / not preferred over the other.

```
pthread_t tid;
int i;
void *pointer = &i;

pthread_create(&tid, NULL, foo, (void *)i);
pthread_create(&tid, NULL, foo, pointer);
```

(n) Construct process/thread graph for the following code snippet.

```
void *mythread(void *vargp) {
  printf("First line in thread\n");
  printf("Second line in thread\n");
  return NULL;
}
int main() {
pthread_t tid;
printf("First line in main thread\n");
pthread_create(&tid, NULL, mythread, NULL);
printf("Second line in main thread\n");
pthread_join(tid, NULL);
printf("Third line in main thread\n");
printf("Fourth line in main thread\n");
exit(0);
}
```

Count the total number of processes and peer threads based-on your process/thread graph when the above code is executed.

(o) Demonstrate different ways of thread termination using suitable code snippet.
(p) Demonstrate the use of network byte order and host byte order in a program using suitable system / library calls.
(q) Differentiate ULT and KLT using a suitable diagram.
(r) Demonstrate the concept in question (d) using a suitable code snippet. Highlight only important parameters to various system / library calls without getting into the exact syntax / function prototypes.
(s) send() / recv() calls could be replaced with sendto() /recvfrom() calls, and vice versa. Comment (3 to 4 sentences) on the correctness of this statement with suitable example.
(t) Demonstrate the difference between a default signal handler and a user-defined signal handler using a suitable code snippet.

2. Demonstrate the working of Peterson's solution for process synchronization clearly identifying example scenarios to prove that it,
   (a) does not violate mutual exclusion
   (b) avoids / has race condition
   (c) avoids / has starvation
   (d) avoids / leads to deadlock
   (e) has / no busy-waiting
   (Do not write any theory or code / pseudocode (as given in notes / books) of Peterson's solution in your answer. You could use your own code / pseudocode to create and demonstrate example scenarios.) [5]

3. Demonstrate the use of pipes using a suitable flowchart to implement a producer-consumer problem. Also, clearly show parts of your implementation / code wherever pipes are being used by the producer and consumer processes. Will pipes be useful in addressing the requirements of a dining philosopher problem? If yes, how? If not, what are the difficulties? Give code snippet in support of your reasons. [5]

4. Design a simple client server application where a server receives a messages from client A, then sends an acknowledgement to client A as well as the message received from client A to client B. When client B receives the message from the server, it sends a new message to the server, which then acknowledges back along with sending the message received from client B to client C. A similar message exchange happens between client C and client A through the common server. Server is listening to all the clients on port 5555. Client does not terminate after sending the message to the server, however, is ready to send the next message after it receives the acknowledgement from the server for the previous message. Write a pseudocode for the above client server application using suitable system / library calls. Highlight only important parameters to various system / library calls without getting into the exact syntax / function prototypes. Also, show important scenarios about how will you test this application to demonstrate the required working. You could also use suitable block diagrams to show the working of such an application. Assume that clients and server use TCP. [5]