Soumyadip Pyne 197178 CSE A

1.

No. of Processes ⇒ 6

Name of Processes ⇒ ./performer ./screen ./anchor
./j1 ./j2 ./j3

IPCs used ⇒ Socket (for showing performance and announcement of result)
, message-queue (for question answering and putting score

Special BSD calls → recv, send, sendmsg, recvmsg, socket, bind, listen, connect, accept-

I/o Mux → Poll

Signals → Yes.

Name → sig1-handler in ~~Scheduler~~ A for → S notifying A

sig1-handler in judge for → A alerting judge

sig-1 handler in P for → A notifying judge is ready.

FD passing → no explicit need.
(unless P shares to A for announcement).

steps

1. all the mentioned process are active
2. Performer connects to screen using socket.
3. Performer sends data through socket and screen shows it (assume std-out)
4. Performer sends a certain phrase like "over" to notify Screen that performance is over.
5. Screen uses sigusR1 to notify Anchor who has special handler. (screen finds pid of anchor from name)
6. Anchor randomly picks judge and notifies it through sigusR1.
7. Anchor puts judges pid in msg-queue and notifies P through sigusR1 to read it.
8. P reads the pid of judge and communicate through msg que.
9. P sends answer to screen which it displays.
10. Judges put the scores in msg-queue.
11. When A knows the scores from all judges it shows it on screen either by.
    i) explicitly connecting to socket (used)
    ii) or by getting sfd from P.

Server

```
int main()
{
    // socket creation, setsockopt(), bind(), listen()
    p_mfd = accept(sfd, (struct sockaddr*)&addr, (socklen_t*)
                                        &sizeof(addr));
    a_mfd = accept( " ) //similar as above;

    struct pollfd pfds[2]
    for(int i=0; i<2; i++)
        pfds[0].fd = p_mfd;
        pfds[0].events = POLLIN;

        pfds[1].fd = a_mfd;
        pfds[1].events = POLLIN;

    int a_fd = fileno(popen("pidof ./anchor"
                            , "r");
    char s[1024];
    read(fd, &s, 1024);
    pid_a = atoi(s);
while(1) {
    int ret = poll(pfds, 2, 60*30);
    for(int i=0; i<2; i++) {
        if ( pfds[i].revent & POLLIN)
        { int rr = recv(pfds[i], s, 1024);
          if (s == "com")
              kill( pid_a, SIGUSR1);
```

Bhowmik.

```
else
    printf("%s\n", s);
}
```

For Anchor

```
int main
{
    int n = random();
    int fd = fileno(popen("pidof ./judge\n", "r"));
    char s[1024];
    read(fd, &s, 1024);
    judge_pid = atoi(s);
// about judge msg sigusr1
// msg queue init → ftok(), msgget();
// put judge_pid using msgsnd();
// notify p using sigusr1
// reach msgqueue for three judges to
   check score using msgrcv();
// when read all 3, send to screen using
   send(sfd, ....);
}
```

Soumyadip Sengupta 197178 CSE A

Judge

```
int main()
{
    // msg-que init → ftok(), msgget(),

    // alerted by sigUSR1 start msgsnd() with
       question.

    // finds scores in msg-queue()
}
```

Sengupta.