|  |
| --- |
| **[514126] 컴퓨터네트워크** |
| **실습 #06 문제 및 보고서** |

|  |  |
| --- | --- |
| **이름** | 정수현 |
| **학번** | 20185290 |
| **소속**  **학과/대학** | 전자공학과 |
| **분반** | 02 (담당교수: 박찬영) |

<주의사항>

* 개별 과제 입니다. (팀으로 진행하는 과제가 아니며, 모든 학생이 보고서를 제출해야 함)
* 각각의 문제 바로 아래에 답을 작성 후 제출해 주세요.
* 소스코드/스크립트 등을 작성한 경우, 해당 파일의 이름도 적어주세요.
* SmartLEAD 제출 데드라인:
* 다음 다음 실습시간 전날 23:55까지 (2주간 진행하는 과제 입니다)
* 데드라인을 지나서 제출하면 0점
* 주말/휴일/학교행사 등으로 인한 데드라인 연장 없음
* 부정행위 적발 시, 원본(보여준 사람)과 복사본(베낀 사람) 모두 0점 처리함
* SmartLEAD 에 아래의 파일을 제출해 주세요
* 보고서(PDF 파일로 변환 후 제출을 권장하나, WORD로 제출해도 됨)
* 보고서 파일명에 이름과 학번을 입력해 주세요.
* 소스코드, 스크립트, Makefile 등을 작성해야 하는 경우, 모든 파일 제출(또는 본 문서에 소스코드 화면 캡처해서 붙여넣기)

<개요>

이번 과제는 리눅스 프로그래밍 Part 2입니다. 다중 프로세스 및 다중 스레드를 생성하는 내용으로 구성되어 있습니다.

<실습 과제>

|  |
| --- |
| **[Q 0] 요약 [배점: 10]**  이번 과제에서 배운 내용 또는 과제 완성을 위해서 무엇을 했는지 2~3문장으로 요약하세요. |

답변: fork()의 쓰임과 다중 스레드를 사용하는 방법을 알았고 stop-and-wait 프로토콜을 직접 구현하여 프레임 오류가 없을 때와 있을 때 어떤 차이를 두어야 하는지에 대해 숙지했습니다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **[Q 1] 다중 프로세스 프로그램 [배점: 20]**  아래의 코드를 실행하면 부모 프로세스는 하나의 자식 프로세스를 생성합니다. 코드에는 총 5번의 printf 구문이 사용되고 있습니다. 아래의 코드(multiprocess.c)를 실행하고, 다음의 질문에 답하세요.   |  |  | | --- | --- | | **코드** | **설명** | |  | ‘fork()’를 호출하는 순간 자식 프로세스가 생성됨.  생성된 자식 프로세스는 ‘fork()’를 호출하는 구문 직후부터 이어서 실행함.  부모 프로세스와 자식 프로세스는 동일한 코드를 실행하지만, 각자 pid 변수에 저장하고 있는 값이 서로 다르기 때문에, switch-case 구문에 의해 서로 다른 코드를 실행하게 됨. 자식 프로세스는 pid 변수에 0을, 부모는 pid에 자식 프로세스 ID를 저장. |     문제 1) 자식 프로세스는 총 몇번의 printf 를 실행하나요?  문제 2) 부모 프로세스는 총 몇번의 printf 를 실행하나요?  문제 3) 각 printf 구문이 출력하는 문자열의 시작 부분에는 “(C1)”, “(P1)”과 같은 prefix가 있습니다. 위 프로그램을 실행하고, 어떤 순서로 문자열이 출력되는지 그 순서를 답하세요. 각 문자열 전체를 입력할 필요는 없고, printf로 출력되는 문자열의 prefix 만 답하면 됩니다. |

답변 1)

2번

답변 2)

3번

답변 3)

(P1) (C1) (C2) (P2) (P3)



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **[Q 2] thread create & join 사용해서 다중 스레드 생성 및 자원 회수 [배점: 20]**  아래의 코드를 실행하면 main 함수 내에서 하나의 스레드가 추가로 생성되고, 생성된 스레드는 start\_routine 이라는 함수를 실행한 뒤 종료합니다. 코드를 실행하고, 아래의 문제에 답하세요.   |  |  | | --- | --- | | **코드** | **설명** | |  | ‘main’ 함수에서 스레드를 생성할 때 start\_routine을 실행하라고 지정하고 char\* msg를 인자로 전달함  스레드가 생성될 때 실행하게 될 함수는 여기 예제에서 보이는 것과 동일한 형식으로 정의해야 함(즉,ca 리턴 타입, 입력 인자값 등을 동일하게 코딩해야 함. 단, 함수 이름은 변경 가능) |     문제 1) ‘main’에서 ‘pthread\_create’를 이용해 생성한 스레드는 총 몇번의 printf 를 실행하나요?  문제 2) 각 printf 구문이 출력하는 문자열의 시작 부분에는 “(MAIN 1)”, “(TH1)”과 같은 prefix가 있습니다. 위 프로그램을 실행하고, 어떤 순서로 문자열이 출력되는지 그 순서를 답하세요. 각 문자열 전체를 입력할 필요는 없고, 문자열의 prefix 만 답하면 됩니다.  \*\* 컴파일 오류 발생 시, “-lpthread” 옵션을 주고 컴파일을 다시 시도해 보세요. |

답변:

답변 1)

3번

답변 2)

(MAIN1) (MAIN2) (TH1) (TH2) (TH3) (MAIN3) (MAIN4)



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **[Q 3] stop-and-wait 프로토콜 + 프레임 변형/분실 오류가 없는 환경 [배점: 30]**  다중 스레드 프로그래밍을 이용해서 간단한 네트워크 시뮬레이션 프로그램을 구현하고, 정지-대기(stop-and-wait) 프로토콜을 아래와 같이 구현하세요.  <필수 요구조건>   |  |  | | --- | --- | | * ‘main’ 함수에서 ‘sender\_routine’ 이라는 함수를 실행하는 sender용 스레드를 생성 (송신 호스트 역할을 하는 스레드) * ‘main’ 함수에서 ‘receiver\_routine’ 이라는 함수를 실행하는 receiver 용 스레드 생성 (수신 호스트 역할을 하는 스레드) |  | | * 송신 호스트용 스레드와 수신 호스트용 스레드는 서로 Frame 과 ACK를 주고받음 (구현 시, 실제로 데이터를 주고 받도록 구현할 필요는 없고 순서 번호를 교환하는 방식으로 구현해도 됨) * 일정 시간 동안 프로그램을 구동한 후(=시뮬레이션 한 후), 터미널 출력 결과는 다음과 같아야 함. ‘(S)’는 sender를, ‘®’은 receiver를 의미함. |  |   <참고>  ‘필수 요구조건’을 제외한 나머지 부분은 자유롭게 구현해도 됩니다. 제가 구현한 코드의 일부는 문서 하단에 첨부되어 있습니다. 필요한 경우 참고하세요.  문제) 일정 시간 동안 프로그램을 실행하고, 터미널 출력 화면을 캡처해서 아래에 첨부하세요 |

답변 )





|  |
| --- |
| **[Q 4] stop-and-wait 프로토콜 + 프레임 변형 오류가 있는 환경 [배점: 20]**  위의 [Q 3]에서 작성한 코드를 확장해서, 확률적으로 프레임 변형 오류가 있는 환경을 구현하세요. 프레임 분실 오류는 없다고 가정하겠습니다 (즉, 타이머 등의 기능을 구현할 필요 없음).  <필수 요구조건>   * 확률적으로 Frame 또는 ACK 수신에 오류가 있는 환경을 재현해야 합니다. * C언어에서 rand() 함수 또는 다른 난수생성함수를 사용해서 확률적인 상황을 재현하세요. * 일정 시간 동안 프로그램을 실행한 경우, 아래와 같은 결과가 출력되어야 합니다. 단, 확률 값에 따라서 ERROR 가 발생하는 시점/빈도수 등등은 달라질 수 있습니다.      * 프레임 변형 오류가 발생할 확률을 30% ~ 50% 사이의 값으로 설정하세요.   문제 1) 확률적으로 프레임 오류가 발생하는 상황을 어떻게 구현했는지, 구현 방식을 간단히 설명하세요.  문제 2) 일정 시간 동안 프로그램을 실행하고, 터미널 출력 화면을 캡처해서 아래에 첨부하세요 |

답변 1)

오류가 발생할 확률을 30~50사이의 랜덤값으로 probNum(오류 발생 확률 변수)에 rand() % (50 - 30 + 1) + 30을 넣어주고, 1~100 사이의 랜덤값 x를 구해서 x가 probNum보다 작거나 같을 경우에 sender에서는 ackFrameID에 -1을 넣고, receiver에서는 sendFrameID에 -1을 넣어 오류를 발생시켰습니다.

답변 2)







**<참고: stop-and-wait + 프레임 변형/분실 오류 없는 환경 구현 예>**

*\*\* 반드시 이렇게 구현해야 하는 것은 아니며, 참고 자료로만 사용하면 됩니다.*

|  |
| --- |
| #include <pthread.h>  #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <sys/types.h>  #include <time.h>  /\*  **전송 매체 역할을 할 구조체 정의:**  이 구조체는 sender와 receiver가 공유해서 사용한다. 스레드와 함께 동시에 실행되는 함수에 공통 인자로 전달.  예를 들어, sender가 sendFrameID=7 이라고 설정하면, 7번 정보 프레임을 전송하고 있다는 의미.  Receiver는 공유하여 사용하는 구조체에서 sendFrameID에 저장된 번호가 7번인 것을 확인하면, 7번 Frame을 정상적으로 받았다고 가정한다. 다음으로, Receiver는 7번 정보 프레임에 대해서 응답을 보내기 위해, ackFrameID=7로 설정한다.  Sender는 ackFrameID가 7이라는 것을 확인하면, 7번 정보 프레임이 정상적으로 수신되었다고 판단하고, 다음 8번 프레임을 전송하기 위해 sendFrameID=8로 설정한다. Receiver는 내부적으로 frameToReceive라는 변수를 사용하는데, 8번 프레임을 받기 전에 frameToReceive는 8로 변경되어 있고, 8번 프레임을 받으면 ackFrameID=frameToReceive로 설정하고 (=즉, 현재 수신한 정보 프레임의 순서 번호와 동일한 번호로 ACK를 전송하고), 다음으로 frameToReceive를 +1 한다.  가상의 환경에서는 sender와 receiver가 동시에 전송을 해도 오류가 없다고 가정  \*/  struct channel {  int sendFrameID;  int ackFrameID;  };  const int const NULL\_FRAME\_ID = -1; // 유효하지 않은 프레임 번호  const int const INITIAL\_FRAME\_ID = 0; // 순서 번호는 0번 부터 시작  const int const TIME\_STEP = 2; // 초단위  const int const ITER\_MAX = 5; // 송수신 절차를 몇 번 반복 할지  void \*sender\_routine(void \*arg){ // 송신 호스트 역할을 할 스레드가 실행하는 함수  struct channel \*chan = (struct channel \*) arg; // type casting  int frameToSend = INITIAL\_FRAME\_ID; // 이번에 전송할 정보 프레임의 순서번호  for( int i = 0 ; i < ITER\_MAX ; i++ ) {  // 정보 프레임 전송 ㄱㄱ  ...    // ACK 프레임 수신  ...  }  pthread\_exit(0);  }  void \*receiver\_routine(void \*arg){ // 수신 호스트 역할을 할 스레드가 실행하는 함수  struct channel \*chan = (struct channel \*) arg; // type casting  int frameToReceive = INITIAL\_FRAME\_ID; // 이번에 수신하기를 기대하는 정보 프레임의 순서번호  sleep((int)(TIME\_STEP / 2));  for( int i = 0 ; i < ITER\_MAX ; i++ ) {  // 정보 프레임 수신  ...    // ACK 프레임 전송  ...  }  pthread\_exit(0);  }  int main(void){  pthread\_t sender;  pthread\_t receiver;  struct channel chan; // 채널 역할을 대신 할 구조체  chan.sendFrameID = NULL\_FRAME\_ID; // 유효하지 않은 순서번호로 초기화  chan.ackFrameID = NULL\_FRAME\_ID; // 유효하지 않은 순서번호로 초기화  pthread\_create(&sender, NULL, sender\_routine, (void\*) &chan);  pthread\_create(&receiver, NULL, receiver\_routine, (void\*) &chan);  pthread\_join(sender, NULL);  pthread\_join(receiver, NULL);  return 0;  } |
| 코드 중간 중간에 sleep 함수를 넣어줘서, sender와 receiver 가 순서에 맞게 동작할 수 있도록 코딩했습니다.  예)  [sender] 정보 프레임 전송 => sleep => ACK 프레임 수신 => ...  [receiver] 정보 프레임 수신 => ACK 프레임 전송 => sleep => ... |

**끝! 수고하셨습니다**