Real-Time Systems & Lab

Assignment #2

- **0 Score** for the corresponding assignment **for both the copy and source**.
- **Clearly indicate** the source if you get some from other places or if it is not your original idea or thoughts. **(Otherwise, the score will be cut by 10%)**

1. **Fill in the blanks.**

Real-Time system is a system operating under (**temporal**) (**constraints**)

2. **What is a proper matching word below?**

Hard RTS -> **Safety-critical**

Soft RTS -> Inconvenience

3. **What are the main requirements in RTS?**

실시간 시스템(RTS)에서 일반적으로 필요한 주요 요구사항은 기능적 요구사항(Functional requirements), 시간적 요구사항(Temporal requirements), 신뢰성 요구사항(Dependability requirements)의 3 가지이다.

기능적 요구사항 (Functional requirements)

실시간 시스템의 기능적 요구사항은 시스템의 핵심 작동 방식과 그 기능에 관한 중요한 기준들을 정의한다.

- **데이터 수집:** 실시간 시스템의 변수들을 연속적 및 이산적으로 샘플링하여 실시간 데이터베이스에 반영하며, 각 변수의 값에 변화가 있을 때마다 즉각적으로 데이터베이스를 업데이트한다.
- **디지털 직접 제어:** 센서와 액츄에이터 등에 직접적으로 접근하여 실제 제어를 가능하게 한다.
- **사용자와의 상호작용:** 시스템의 현재 상태를 사용자에게 정보로 제공하고, 필요한 로그와 경고 메시지를 통해 시스템의 정상 작동을 지원한다.

시간적 요구사항 (Functional requirements)

실시간 시스템의 시간적 요구사항은 제어되어야 할 과정의 물리적 역학 관계에 따라 발생한다.

- **시스템 관찰 지연:** 시스템 상태의 관찰에 필요한 지연 시간
- **제어 값 계산 지연:** 새로운 제어 값의 계산에 필요한 지연 시간
- **이전 지연으로부터의 변동 (jiter):** 이전 지연에 대한 변동성

신뢰성 요구사항 (Dependability requirements)

실시간 시스템은 사람들의 생명과 안전, 경제적 영향 및 손실이 큰 중요한 분야에서 활용된다. 이러한 중요성으로 인하여, 실시간 시스템은 뛰어난 안정성을 필요로 한다.

- **높은 신뢰성(High Reliability):** 경성 실시간 시스템(Hard Real-Time System)과 같은 시스템은 고도의 신뢰성 요구사항을 가진다. 이러한 시스템의 실패율은 시간당 10^{-9}회 미만을 만족해야 한다.
- **안정적인 인터페이스 (Stable interfaces):** 중요한 서브시스템과 그 외의 서브시스템 사이에는 안정적인 인터페이스가 필요하다. 이는 서로 간의 오류 전파를 방지하기 위함이다.
- **독립적 서브시스템 구조:** 전체 시스템 구조는 다른 서브시스템과 독립적으로 검증할 수 있는 속성을 가진 자율적인 서브시스템으로 구성되어야 한다.

4. **Correct output is not enough in RTS. What else do we need?**

시기 적절성 (Timeliness):

실시간 시스템에서는 결과가 단순히 논리적으로 올바르다는 것만으로는 충분치 않다. 그 결과가 올바른 시간 내에 제공되는 것이 중요하다. 결과의 유효성은 그 결과가 언제 제공되는지에 따라 결정된다. 예를 들면, 항공기 제어 시스템에서 제어 명령이 늦게 발송되면 그 결과는 치명적일 수 있다.

신뢰성 (Reliability):

실시간 시스템은 종종 인명이나 경제적 가치에 영향을 줄 수 있는 중요한 애플리케이션에서 사용된다. 이러한 시스템에서는 높은 신뢰성이 요구된다. 특히 안전 중심의 시스템에서는 신뢰성이 매우 중요하며, 잠재적인 위험을 최소화하기 위해 시스템의 오류율과 실패율이 매우 낮아야 한다.

최악의 경우에 대한 시나리오 고려:

실시간 시스템 설계 시에는 최악의 경우를 고려하는 것이 중요하다. 이는 시스템의 성능과 신뢰성을 보장하기 위한 것이다. 예를 들면, 특정 상황에서 시스템의 반응 시간이나 실행 시간에 대한 최악의 경우를 분석하여, 그러한 상황에서도 시스템이 올바르게 동작할 수 있는지 확인해야 한다.

5. **List non-functional constraints in RTS to meet.**

- **타이밍 제약(Timing Constraints):** 실시간 시스템은 정해진 시간 안에 반응하거나 작업을 완료해야 한다.
- **신뢰성(Reliability):** 시스템은 예측 가능한 방식으로 항상 올바르게 작동해야 한다.
- **성능(Performance):** 실시간 시스템은 종종 고성능이 필요하며, 따라서 반응 시간, 처리량 등이 중요하다.
- **보안(Security):** 실시간 시스템은 종종 중요한 미션을 수행하기 때문에, 안

전하게 작동해야 한다.

- **확장성(Scalability):** 몇몇 실시간 시스템, 특히 분산 실시간 시스템에서는 증가하는 데이터 또는 작업 부하에 대응해야 할 수 있다.
- **유지 보수성(Maintainability):** 실시간 시스템도 시간이 지나면서 유지 보수가 필요할 수 있다. 특히 안전 관련 실시간 시스템에서는 유지 보수성이 중요하다.

6. **Give at least three examples for hard and soft deadline real time system.**

경성 실시간 시스템: 마감 시간(Deadline)을 지키지 않는 경우, 심각한 실패나 치명적 결과를 초래할 수 있는 시스템

- **항공기 제어 시스템:** 이 시스템은 비행기의 안정성과 조종을 관리한다. 실시간 작업 지연 발생 시 추락 같은 치명적 결과가 발생할 위험이 있다.
- **자동차의 ABS (Anti-lock Braking System):** 브레이크를 밟을 때 브레이크 잠김을 방지한다. 실시간 응답이 없을 경우 사고 위험이 증가한다.
- **페이스 메이커:** 심장 박동을 제어한다. 지연 발생 시 환자의 생명에 위협이 될 수 있다.
- **산업용 로봇:** 생산 라인에서 물체를 정밀하게 이동하거나 조작하기 위해 실시간 제어가 필수적이다.
- **원자력 발전소 제어:** 냉각 시스템 등 중요한 시스템의 작동을 실시간으로 모니터링하고 제어해야 한다.

연성 실시간 시스템: 마감 시간을 약간 넘겨도 치명적인 실패로 이어지지는 않지만, 성능 저하나 사용자 경험 저하를 초래할 수 있는 시스템

- **비디오 스트리밍:** 일시적인 지연이나 버퍼링이 발생할 수 있다. 이로 인해 사용자 경험이 저하될 수 있으나, 스트리밍은 계속된다.
- **인터넷 전화 서비스 (VoIP):** 통화 중 약간의 지연이나 음성 손실이 발생할 수 있다. 사용자는 일시적 불편을 느낄 수 있으나, 통화는 중단되지 않는다.

- **온라인 게임:** 서버와의 통신 지연으로 인한 일시적 렉이나 반응 지연이 발생할 수 있다. 게이머는 불편을 느낄 수 있으나, 게임은 계속된다.
- **DVD 플레이어:** 노래나 영상 재생 중 일시적인 지연이나 버퍼링이 발생할 수 있다. 사용자는 일시적 불편을 느낄 수 있으나, 재생은 계속된다.

7. **What is the difference between Digital Twin Systems and CPS?**

디지털 트윈 시스템: 실제 물리적 대상의 가상 복제본을 만드는 시스템으로, 실세계의 대상을 시뮬레이션, 분석, 예측하기 위해 사용된다.

사이버-물리 시스템(CPS): CPS는 사이버(계산, 통신, 제어) 부분과 물리적(자연 및 인간이 만든 시스템) 부분이 모든 규모와 수준에서 긴밀하게 통합된 시스템이다.

차이점 – 적용 범위 측면:

디지털 트윈 시스템: 특정 요소나 시스템을 디지털 환경에서 복제하는 것에 중점을 둔다.

사이버-물리 시스템(CPS): 물리적 요소와 계산 서비스를 합친 보다 광범위한 시스템을 포함한다.

차이점 – 상호 작용성 측면:

디지털 트윈 시스템: 실제 대상에는 직접 영향을 주지 않지만 디지털 트윈은 실제 세계에서의 데이터를 기반으로 조정된다.

사이버-물리 시스템(CPS): 가상 계산이 물리 세계에 실시간 영향을 미치고 그 반대의 상호 작용이 있는 양방향 상호 작용을 포함한다.

차이점 – 응용 측면:

디지털 트윈 시스템: 제품 설계, 최적화, 유지 보수 예측 등의 산업에서 널리 사용된다.

사이버-물리 시스템(CPS): 실시간 제어와 물리적 세계와의 상호 작용이 필

수적인 자동화, 로봇 공학, 스마트 그리드, 스마트 도시 및 건강 모니터링 시스템에서 흔하다.

8. **Compare between General-Purpose Systems and Real-time systems.**

범용 시스템 (General-purpose systems): 주로 공정성과 전체적인 throughput 을 최적화하려고 설계되었다. 평균적인 성능과 다양한 작업의 효율적인 처리가 중요하다.

실시간 시스템 (Real-time systems): 주로 특정 작업을 연관된 마감 시간 내에 완료하는 데 초점이 맞춰져 있다. 마감 시간 준수는 극단적으로 중요하며, 준수하지 않을 경우 심각한 결과를 초래할 수 있다.

성능 지표 측면:

범용 시스템: 일반 시스템의 주요 성능 지표에는 평균 응답 시간, 시스템 throughput, CPU 이용률이 포함된다. 균형 잡힌 성능을 제공하고 광범위한 응용 프로그램을 효과적으로 제공하는 것이 목표다.

실시간 시스템: 성능은 주로 시스템의 마감 시간 준수 능력으로 측정된다. 실시간 시스템에서 다루는 관심 지표에는 작업 완료 시간의 변동 (jitter), 최장 실행 시간(Worst-case execution time)등이 포함된다.

작업 우선 순위 측면:

범용 시스템: 공정성이 우선시된다. 이는 시스템이 모든 작업에 차례를 보장하기 위해 노력한다는 것을 의미한다. 이는 기아 상태(starvation)을 방지하기 위해 라운드 로빈 스케줄링이나 에이징 기법(Priorty Aging)과 같은 메커니즘을 사용할 수 있다는 것을 의미한다.

실시간 시스템: 작업은 그 긴급성이나 중요성에 따라 우선 순위가 결정된다. 많은 실시간 시스템에서 작업은 마감 시간, 중요성 또는 기타 요소를 기반으로 정적 또는 동적 우선 순위가 부여된다. 공정성이나 균형

잡힌 리소스 할당에 대한 관심보다 중요한 작업이 마감 시간 내에 완료되도록 보장하는 것이 종종 중요하다.

사용 사례 측면:

범용 시스템: 일반적인 응용 프로그램에는 데스크톱 운영 체제, 일반 목적의 서버, 엄격한 시간 요건 없이 다양한 작업을 처리하기 위해 설계된 시스템이 포함된다. 예로 웹 브라우징, 문서 편집, 멀티미디어 재생이 있다.

실시간 시스템: 시간 내에 작업 완료가 필수적인 맥락에서 사용된다. 예로 자동차의 임베디드 시스템(에어백 배포, 엔진 제어), 의료 장비(심장 박동기), 항공 통제 시스템, 산업 자동화 시스템 등이 있다.

9. **List features for efficiency in RTOS.**

- **결정론적 동작(Determinism):** 실시간 운영체제는 작업이 예측 가능한 방식으로 예정된 시간 안에 스케줄링 및 완료될 것이라는 것을 보장할 수 있다.
- **높은 성능:** 실시간 운영체제는 빠르고 반응성이 좋게 설계된다. 이는 작업이 엄격한 마감 시간 내에 완료되어야 하는 실시간 시스템에서 중요하다.
- **적은 메모리 사용량:** 실시간 운영체제는 일반적으로 경량화되어 있으며 적은 메모리 사용량을 가진다. 이는 종종 제한된 리소스를 가진 임베디드 시스템에서 중요하다.
- **우선순위 기반 스케줄링:** 실시간 운영체제는 가장 중요한 작업이 먼저 실행되도록 우선순위 기반 스케줄링(Priorty Scheduling)을 사용한다.
- **자원의 효율적 사용:** 실시간 운영체제는 사용 가능한 리소스, 예를 들면 CPU 시간, 메모리, 주변 장치 등을 효율적으로 사용하도록 설계된다.
- **선점형 멀티태스킹(Preemptive multitasking):** 현재의 작업이 완료되지 않았더라도 실시간 운영체제는 작업 간 전환을 할 수 있는 방식으로 설계된다.

10. **Why isn’t Linux real-time?**

리눅스는 일반적인 용도의 운영체제로 설계되었기 때문에, 기본적으로 일정한 시간 내에 실행될 것을 보장하지 않는다.

리눅스가 실시간 운영체제가 아닌 이유는 다음과 같다:

시스템 호출 비선점: 리눅스는 시스템 호출을 선점하지 않는다. 따라서 낮은 우선 순위의 프로세스가 시스템 호출을 실행하는 동안 높은 우선 순위의 프로세스가 실행을 기다릴 수 있다. 이로 인해 높은 우선 순위의 프로세스가 마감 시간을 놓칠 수 있다.

비정적 메모리 관리 시스템 사용: 리눅스는 동적 메모리 관리 시스템을 가지고 있다. 이는 커널이 언제든지 메모리 페이지를 이동할 수 있다는 것을 의미한다. 이는 실시간 작업에 대한 예측 불가능한 지연을 초래할 수 있다.

스케줄링 정책: 리눅스는 프로세스와 작업 스케줄링을 위해 완전 공정 스케줄러(Completely Fair Scheduler, CFS)를 사용한다. CFS 는 일반적인 용도에 대한 좋은 성능과 공정성을 제공하지만, 실시간 시스템에서 필요한 엄격한 시간 제약을 보장하지 않는다.

11. **Explain First-In-First-Out, Round-Robin and Earliest Deadline First scheduling.**

First-In-First-Out (FIFO):

정의: FIFO 스케줄링은 프로세스가 도착한 순서대로 처리하는 방식이다.

특징: 구현이 간단하며 예측이 쉽다. 그러나 짧은 작업이 긴 작업 뒤에 오면 긴 시간 동안 기다려야 할 수 있다.

작동 방식: 프로세스가 도착하는 순서대로 큐에 삽입되고, 큐의 맨 앞에 있는 프로세스가 CPU를 할당받아 실행된다.

예시: A, B, C 세 프로세스가 도착 순서대로 대기 중이면, A가 완료될 때까지 B와 C는 기다려야 한다.

Round-Robin (RR):

정의: 모든 프로세스에게 동일한 양의 CPU 시간 할당을 주기적으로 제공하는 방식이다.

특징: 모든 프로세스에 고르게 CPU 시간을 분배한다. 타임 퀀텀 선택에 따라 성능이 달라질 수 있다.

작동 방식: 프로세스들에게 순차적으로 타임 퀀텀만큼의 실행 시간을 할당한다. 타임 퀀텀이 지나면 다음 프로세스로 넘어간다.

예시: A, B, C 세 프로세스가 있고, 타임 퀀텀이 10ms라면, A -> B -> C 순으로 10ms씩 실행된다.

Earliest Deadline First (EDF):

정의: 각 프로세스의 마감 시간을 기준으로 가장 빠른 마감 시간을 가진 프로세스부터 처리하는 방식이다.

특징: 마감 시간이 긴급한 작업을 우선적으로 처리한다. 일부 작업의 마감 시간이 계속 초과될 경우 해당 작업이 배제될 위험이 있다.

작동 방식: 프로세스들의 마감 시간을 확인하여 가장 빠른 마감 시간을 가진 프로세스가 CPU를 할당받아 실행된다.

예시: A, B, C 세 프로세스의 마감 시간이 각각 5ms, 15ms, 10ms라면, A -> C -> B 순서로 실행된다.

<References>

1. Real-Time Scheduling Lecture Notes - Washington University
2. EECS571 Lecture Notes - The University of Michigan
3. Rea-Time Systems Lecture Presentation - University of Aveiro
4. Real-Time Operating System (RTOS): Types, Examples & Features: https://www.prepbytes.com/blog/operating-system/what-is-a-real-time-operating-system-rtos/
5. What is a Real-Time Operating System (RTOS)? - NI: https://www.ni.com/en/shop/data-acquisition-and-control/add-ons-for-data-acquisition-and-control/what-is-labview-real-time-module/what-is-a-real-time-operating-system--rtos--.html
6. Real Time Operating Systems - Teach Computer Science: https://teachcomputerscience.com/real-time-operating-systems/
7. Earliest Deadline First (EDF) CPU scheduling algorithm - GeeksforGeeks: https://www.geeksforgeeks.org/earliest-deadline-first-edf-cpu-scheduling-algorithm/
8. Difference between Hard real time and Soft real time system - GeeksforGeeks: https://www.geeksforgeeks.org/difference-between-hard-real-time-and-soft-real-time-system/
9. Functional vs Non Functional Requirements - GeeksforGeeks: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/
10. RTLinux - Wikipedia: https://en.wikipedia.org/wiki/RTLinux
11. Digital twin - Wikipedia: https://en.wikipedia.org/wiki/Digital_twin
12. 디지털 트윈(Digital Twin)이란? - IBM: https://www.ibm.com/kr-ko/topics/what-is-a-digital-twin
13. 디지털 트윈의 기술요소 및 CPS - Tistory: https://slaks1005.tistory.com/19
14. 우선순위 스케줄링(Priority Scheduling) - Tistory: https://jhnyang.tistory.com/132
15. What Is A Real-Time Operating Systems (RTOS) - Wind River: https://www.windriver.com/solutions/learning/rtos

16. Is Linux truly real time? - Quora: https://www.quora.com/Is-Linux-truly-real-time-1

17. Real-time operating system (RTOS): Components, Types, Examples - GURU99: https://www.guru99.com/real-time-operating-system.html

<The End of the Assignment>