클립아트, 만화 영화이(가) 표시된 사진

자동 생성된 설명

*COSE341(01)*

*Operating system*

*Project #3 Report*

Your name: Yeyoung Park

Student #: 2020160027

YOUR DEPARTMENT: Mathematics

Submission date: 2024/12/05

Number of free days used for this project: 0 day

# Development environment

* 1. Host OS : windows 10
  2. VirtualBox : version 7.1.2
  3. Ubuntu : 18.04.2 LTS
  4. Linux Kernel : 4.20.11.sslab

# Descriptions for memory analysis [20 pts]

1. Describe physical memory, virtual memory, and shared memory, respectively (5 pts)

Physical memory는 컴퓨터에서 실제로 설치된 RAM(Random Access Memory)를 의미한다. CPU가 데이터를 처리할 때 일반적으로 직접 접근할 수 있는 가장 빠른 저장 공간이기 때문에, SSD나 HDD같은 저장 공간에 저장된 Program을 실행할 때 SSD나 HDD에서 Instruction을 직접 하나씩 접근하는 것이 아니라 OS가 RAM으로 적절히 Program을 옮겨서 실행시킨다. OS는 이를 Process별로 관리하며, Physical memory는 한정되어 있기 때문에 각 Process가 Physical memory의 특정 부분을 잘 활용하여 남는 공간이 없이 사용할 수 있도록 할당한다.

Virtual memory는 각 Process가 독립적인 Virtual address space를 보게 하는 기술이다. 이 가상 주소 공간은 OS가 Physical memory와 매핑하여 관리한다.

Shared memory는 여러 Process가 데이터를 같이 읽고 쓸 수 있도록 하는 하나의memory공간을 의미한다.

1. Describe the need for virtual memory and shared memory, respectively (5 pts)

Virtual memory는 각 프로세스에 독립적인 주소 공간을 제공함으로써 모든 프로세스가 충분한 메모리를 가진 것처럼 동작하게 한다. 다른 프로그램의 간섭을 고려하지 않고 구현할 수 있어서 개발의 용이성을 제공한다. 또한, 각 프로세스 별로 독립적으로 관리함으로써 다른 프로세스가 해당 메모리에 접근하지 못하도록 막는다. 즉, Protection domain을 제공한다. 또한, Physical memory를 많은 Process를 실행하는데 활용함으로써 낭비되는 공간을 줄이고 Multiprogramming을 가능하게 한다.

Shared memory는 동일한 데이터를 여러 프로세스가 사용하고 있는 경우 개별적으로 관리하는 것이 아니라 같이 관리함으로써 메모리 사용량을 줄이도록 한다. 또한, IPC를 Kernel 단계에서 관리하지 않고 User 단계에서 관리함으로써 Overhead를 줄이고 낭비되는 시간을 줄인다. 즉, 빠르고 효율적인 데이터 교환과 Resource의 절약을 가능하게 한다.

1. Describe the advantages and disadvantages of virtual memory (5 pts)

장점은 위에서 설명한 그대로이다. 단점은 다음과 같다. Virtual memory는 메모리 주소를 변환해야 하므로, MMU와 같은 하드웨어 단계에서의 지원이 필요하다. 이는 memory에 접근하는 연산을 진행할 때마다 거치므로 필수적으로 늘어나는 Overhead이다. 또한, Page table과 같이 OS 단계에서 메모리 mapping을 관리해야 하므로 Software overhead가 발생한다. 디스크를 활용하는 경우 RAM보다 속도가 훨씬 느리며, 만약 Disk swap이 많이 발생하는 경우 Page fault로 인해 성능이 크게 저하될 수 있다. 또한, Page table, TLB와 같은 복잡한 데이터 구조를 구현해야 하므로 OS 설계의 복잡성을 증가시킨다.

1. Describe the advantages and disadvantages of shared memory (5 pts)

장점은 다음과 같다. 프로세스간 데이터를 메모리에서 직접 교환하므로 IPC를 빠르게 할 수 있다. 또한 효율적인 메모리 사용이 가능하여 메모리 자원을 절약하고 시스템 효율성을 높인다. 또한 기존 운영체제에서 제공하는 API(POSIX Shared Memory)를 활용함으로써 비교적 간단하게 구현할 수 있다. 단점은 다음과 같다. 여러 프로세스가 동시에 메모리에 접근하기 때문에 동기화 되지 않은 데이터에서 문제가 생길 수 있다. 즉, 데이터의 일관성이 깨진다. 이를 해결하기 위해 추가적으로 Lock, Semaphore과 같은 동기화 메커니즘들이 필요하며, 이는 구현의 복잡성으로 이어진다. 또한, 데이터를 같이 공유하기 때문에 보안의 문제가 생길 수 있다. 더불어 공유 메모리를 적절히 해제하는 단계를 거치지 않으면 메모리 누수가 생길 수 있다.

# Implementation [20 pts]

### ku\_mem.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

// Function to read memory usage from /proc/[PID]/status

void get\_memory\_usage(int pid) {

char path[40], line[256];

FILE \*status\_file;

long physical\_memory = 0, virtual\_memory = 0, shared\_memory = 0;

// Construct the path to /proc/[PID]/status

snprintf(path, sizeof(path), "/proc/%d/status", pid);

// Open the /proc/[PID]/status file

status\_file = fopen(path, "r");

if (!status\_file) {

perror("Error opening /proc/[PID]/status");

exit(EXIT\_FAILURE);

}

// Read the file line by line

while (fgets(line, sizeof(line), status\_file)) {

if (strncmp(line, "VmRSS:", 6) == 0) { // Physical Memory

sscanf(line, "VmRSS: %ld kB", &physical\_memory);

} else if (strncmp(line, "VmSize:", 7) == 0) { // Virtual Memory

sscanf(line, "VmSize: %ld kB", &virtual\_memory);

} else if (strncmp(line, "RssShmem:", 9) == 0) { // Shared Memory

sscanf(line, "RssShmem: %ld kB", &shared\_memory);

}

}

fclose(status\_file);

// Print the memory usage

printf("Physical Memory current: %ld kB\n", physical\_memory);

printf("Virtual Memory current: %ld kB\n", virtual\_memory);

printf("Shared Memory current: %ld kB\n", shared\_memory);

}

int main(int argc, char \*argv[]) {

if (argc != 2) { // take argument only 2. first is ku\_mem, second is PID

fprintf(stderr, "Usage: %s <PID>\n", argv[0]);

return EXIT\_FAILURE;

}

int pid = atoi(argv[1]);

// Run the memory check every 0.5 seconds

for (int i = 0; i < 120; ++i) { // 1 minute

printf("Memory check for PID: %d\n", pid);

get\_memory\_usage(pid);

printf("------------------------------------\n");

usleep(500000); // Sleep for 0.5 seconds

}

return EXIT\_SUCCESS;

}

# Experiment results & analysis [30 pts]

# \* The listed items are mandatory minimum requirements. You may include additional, diverse analyses beyond these items.

1. Draw three individual line graphs when running Firefox and measuring the physical, virtual, and shared memory usage. (x-axis: time, y-axis: memory usage) (6 pts)

텍스트, 라인, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

텍스트, 도표, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명

텍스트, 도표, 라인, 그래프이(가) 표시된 사진

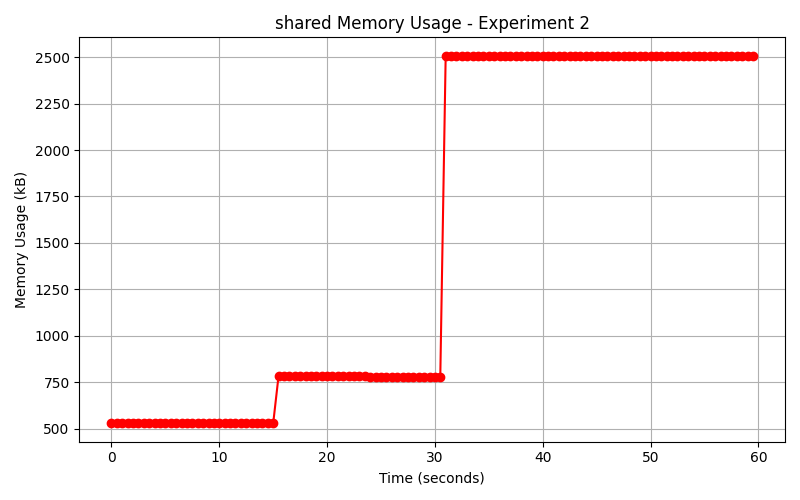
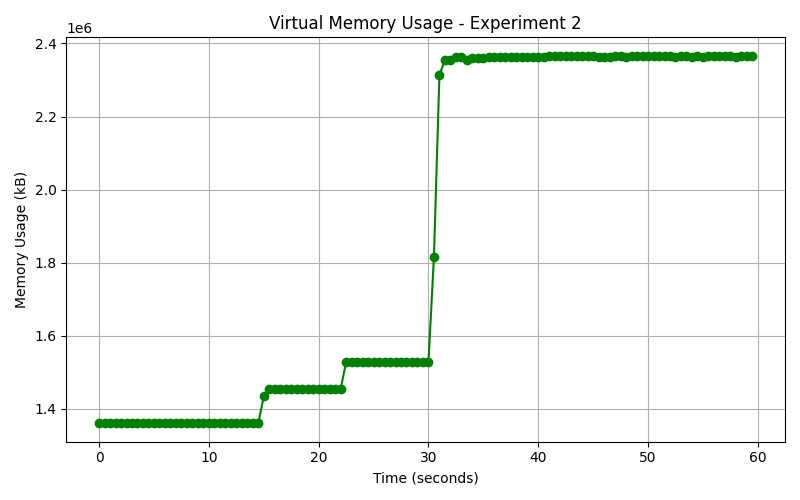
자동 생성된 설명

1. Analyze and explain the changes in memory usage in the graphs drawn in the above (1) (6 pts)

새로운 링크를 클릭한 시점은 8초, 25초, 50초 시점이고, 25초와 50초 시점 이전에는 웹사이트 좌측 상단의 사이드 바를 눌렀다. 그 사이에는 스크롤을 했다. Physical memory 사용량을 보면, 8초이후를 시기로 하여 점점 메모리 사용량이 증가함을 확인할 수 있다. 이후 스크롤을 하며 이미지 로딩과 데이터 캐싱이 이루어져 메모리가 추가로 증가함을 확인할 수 있다. 25초 즈음으로 하여 다른 웹사이트로 이동하면서 다시 메모리 사용량이 급격하게 증가함을 확인할 수 있었다. Virtual memory 사용량을 보면 8초 이후 시점에 급격하게 증가한다. 이는 첫번째 웹사이트에 접속했기 때문에 페이지 로드 과정에서 추가적인 데이터들이 Virtual memory space에 할당되었음이라고 추측하였다. 25초 시기에는 이전보다 소폭 증가했고 이는 이전 데이터가 일부 유지되었기 때문인 것으로 추측했다. 이후 계속 증가 추세를 보이다가 안정화가 되었다. Virtual memory는 Physical memory보다 더 큰 데이터 공간을 활용하므로 증가 폭이 Physical memory에 비해 더 큼을 확인할 수 있었다. Shared memory 사용량 또한 급격하게 증가하는 구간이 비슷한데, 이는 이미지 데이터와 렌더링 프로세스와 관련된 프로세스 등이 다른 프로세스와 공유됨으로 추측된다. 종합적으로, 이미지 로드 및 렌더링, 텍스트 처리, IO 처리(스크롤)로 인해 메모리 사용량이 증가했다고 분석하였다. 또한, 웹사이트 이동 시에 이러한 이유로 급격히 증가하지만 이전 데이터를 유지하며 안정화를 거치려는 모습을 확인할 수 있었다.

1. Draw three individual line graphs by running video player and measuring the physical, virtual, and shared memory usage (6 pts)

텍스트, 라인, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

1. Analyze and explain the changes in memory usage in the graphs drawn in the above (3) (6 pts)

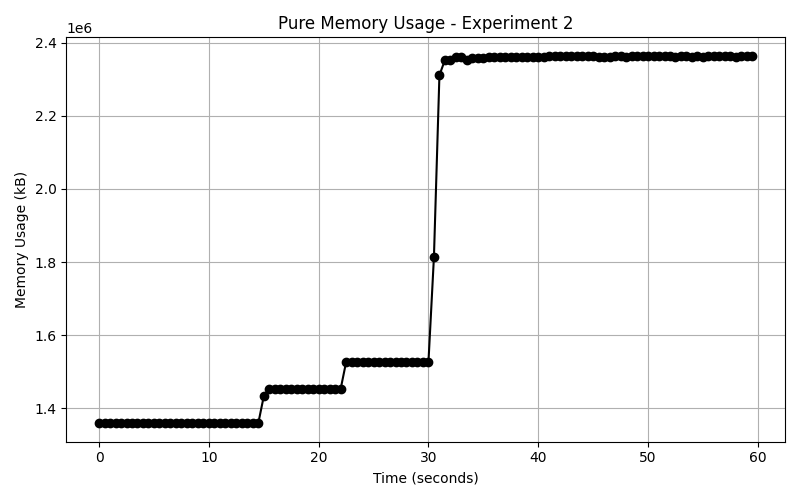
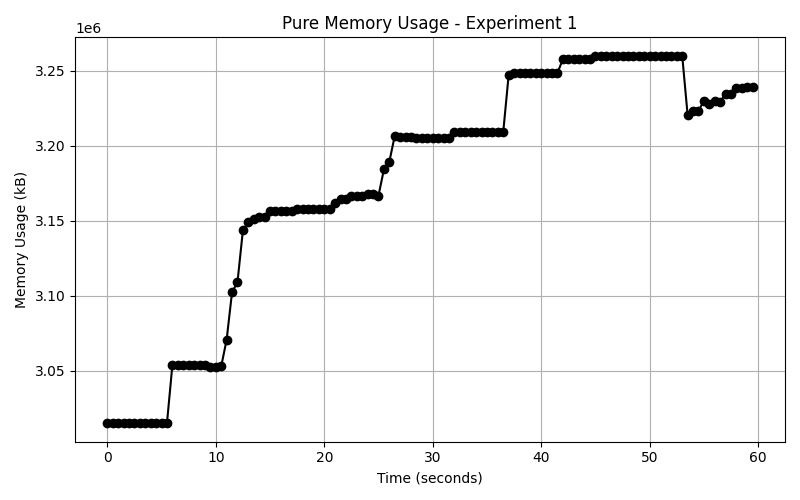
30초 이전의 시점에는 동영상 플레이어에 동영상을 재생하지 않은 상태였고, 30초 시점에 동영상 플레이어에 동영상을 재생했다. Physical memory usage는 30초 이전에는 약 10만 KB의 메모리 사용량을 일정하게 유지하였다. 이 시점에는 동영상을 로드하지 않은 시점이므로 메모리 사용량이 크게 변하지 않았다. 30초 이후, 동영상이 재생되기 시작하면서 급격하게 증가하였다. 20만 KB즈음에서 안정화가 되었다. 메모리 사용량이 급격하게 증가한 원인은 동영상을 재생할 때 필요한 데이터(버퍼)가 메모리에 로드되고, 동영상 플레이어도 재생을 위해 파일을 읽고 디코딩한 데이터를 캐싱하기 때문에 Physical memory의 사용량이 증가했다. Virtual memory의 경우도 급격히 증가했다. 동영상 파일을 읽고 디코딩하기 위해 필요한 추가 작업공간이 Virtual memory space에 할당된다. 주요한 점은 실제 Physical memory보다 증가 폭이 더 크다는 사실이다. 이는 실제 Physical memory(RAM)외에도 디스크에 swap space를 할당해 더 많은 데이터를 처리할 준비를 하기 때문으로 생각된다. Shared memory또한 1750KB정도 경미하게 증가했는데, 이 증가 추세도 급격하다. 동영상 재생 시에 디코더, 렌더링 프로세스 등 여러 프로세스가 공유 데이터를 사용하기 위해 Shared memory space를 사용하므로, 증가했다고 추측된다.

1. Consider which section—stack, data, or code—accounts for most of the changes in physical memory and explain why. (6 pts)

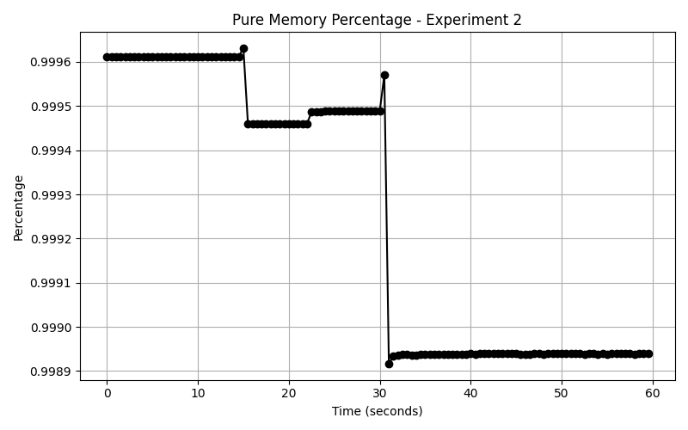
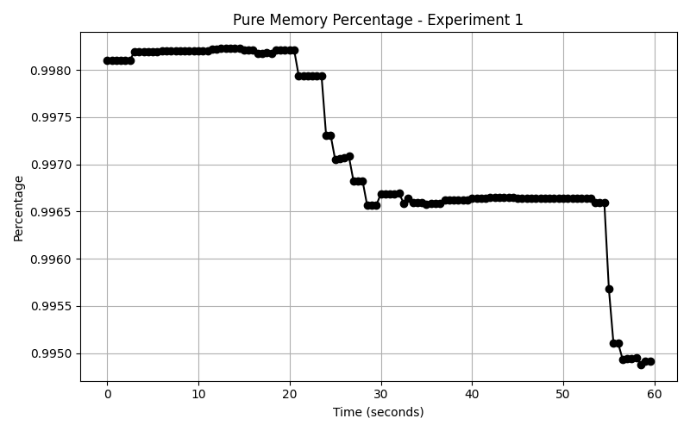
Data영역이 Physical memory영역에서 대부분의 변화를 차지한다고 생각할 수 있다. Stack은 함수 호출, 지역 변수, 함수 반환 주소 등을 저장할 때, Code는 실행 중인 프로그램의 명령어와 함수 코드가 저장되는 영역이며, Data는 전역 변수, 힙, 캐시, 리소스 데이터 등이 저장되는 영역이다. Firefox의 경우 이미지, 텍스트, CSS등의 리소스가 로드되고, 이는 힙에 주로 저장된다. 특히, 이미지가 많은 데이터를 요구하므로 메모리 사용량에 큰 영향을 끼쳤다고 볼 수 있다. Video의 경우 동영상 프레임, 오디오와 같은 데이터를 디코딩하여 저장하며, 버퍼를 관리하는데 많은 메모리를 사용한다. 이 데이터들 또한 힙에서 관리된다. 즉, 프로그램 실행 중 동적으로 할당되는 데이터들 때문에 Physical memory의 대부분이 data영역에서 발생되었다고 생각할 수 있다.

1. Based on the calculated virtual memory and shared memory amounts, analyze and describe the pure memory usage of the process (code + data + stack). (6 pts)

Pure memory usage는 Virtual memory usage에서 Shared memory usage를 빼면 되므로 다음과 같이 나온다.



두 실험 모두 Virtual memory사용량의 경향과 거의 비슷하며, 실제로 Pure memory와 Virtual memory의 비율을 계산하면 다음과 같이 거의 99.99퍼센트의 높은 비율로 Pure memory usage임을 확인할 수 있다.



따라서, Virtual memory = Code + data + stack으로 간주해도 무방하다. 두 실험 모두 공유 메모리를 극히 일부만 사용하고 있기 때문이다.

# Advanced question [10 pts]

# Q. We design a two-level page table structure for given system and scenario. Specify the number of entries in each level of the page table.

Scenario: first-level page table index 10 bits, second-level page table index: 10 bits

System **:**

* Virtual address size: 32 bits
* Physical address size: 28 bits
* Page size: 4KB (4096 bytes)
* Each (last-level) page table entry occupies 4 bytes

# Ans.

Virtual address는 한 프로세스의 페이지 수를 나타내는 Virtual Page Number(이하 VPN)과 각 페이지 별 몇 번째 주소인지를 나타내는 내부 주소 Page offset으로 나뉜다. 이때, Page size는 4KB = 2^12 Byte이므로 Page내부 주소를 나타내기 위해서는 12bits가 필요하다. 다시 말해 Page offset의 크기는 12bits이다. Virtual address size가 32bit이므로 Page offset 12bits를 제외한 20bits는 first-level page table index와 second-level page table index로 각각 10bits씩 나뉘어진다. 이렇게 나누면 first-level page table의 entry는 2^10 = 1024 entry, second-level page table의 entry 또한 2^10 = 1024 entry이다. 총 가상 페이지 수는 2^10 x 2^10 = 2^20 pages가 된다. 만약 이를 single page table로 표현한다고 하면 2^20 x 4bytes = 4MB가 되고, 이를 각 프로세스마다 유지해야 하므로 시스템 메모리를 크게 낭비하게 됨을 확인할 수 있다.