컴퓨터학부 20201853 서민비

1. 개요

1) 메모리 사용량을 출력하는 memsize() 시스템 콜을 구현 및 추가하고 이를 호출하는 쉘 프로그램(memsizetest)을 구현한다.

2)추적할 시스템 콜을 지정하는 정수 mask와 실행할 command를 입력 받아 command가 호출하는 시스템 콜을 추적하는 trace() 시스템 콜과 이를 호출하는 쉘 프로그램(ssu\_trace)을 구현한다.

2. 결과(실행결과 및 설명)

-1)

텍스트이(가) 표시된 사진

자동 생성된 설명

\* Malloc 전, 후 차이가 2048 바이트가 아니라 4096 바이트인 이유는 해제한 매모리 블록을 os에 반환하는 것이 아니라 free list (linked list)에 추가하여 이후 malloc()이 호출되어 요청한 크기를 수용할 수 있을 때 reuse하기 때문이다.

-2)

텍스트이(가) 표시된 사진

자동 생성된 설명

3. 소스코드

user.h

int memsize(void);

int trace(int);

syscall.h

#define SYS\_memsize 22

#define SYS\_trace 23

syscall.c

. . .

extern int sys\_memsize(void);

extern int sys\_trace(void);

static int (\*syscalls[])(void) = {

. . .

[SYS\_memsize]   sys\_memsize,

[SYS\_trace]   sys\_trace,

};

char \*syscallsName[] = {"fork", "exit", "wait", "pipe", "read", "kill", "exec","fstat", "chdir", "dup", "getpid", "sbrk", "sleep", "uptime","open", "write", "mknod", "unlink", "link", "mkdir", "close", "memsize","trace"};

void

syscall(void)

{

  int num;

  struct proc \*curproc = myproc();

  num = curproc->tf->eax;

  if(num > 0 && num < NELEM(syscalls) && syscalls[num]) {

    curproc->tf->eax = syscalls[num]();

    // trace  //

    if (curproc->tracemask >> num) {

      cprintf("syscall traced pid = %d, syscall = %s, %d returned\n", curproc->pid, syscallsName[num-1], curproc->tf->eax);

    }

  } else {

    cprintf("%d %s: unknown sys call %d\n",

            curproc->pid, curproc->name, num);

    curproc->tf->eax = -1;

  }

}

sysproc.c

int

sys\_memsize(void)

{

  uint size;

  //호출한 프로세스의 메모리 사용량을 출력

  size=myproc()->sz;

  return size;

}

int

sys\_trace(void)

{

  int mask;

  if (argint(0, &mask) < 0)

    return -1;

  struct proc \*p=myproc();

  p->tracemask=mask;

  return 0;

}

memsizetest.c

#include "types.h"

#include "stat.h"

#include "user.h"

#define SIZE 2048

int main(void)

{

    int msize=memsize();

    printf(1,"The process is using %dB\n",msize);

    char \*tmp=(char \*)malloc(SIZE\*sizeof(char));

    printf(1,"Allocating more memory\n");

    msize=memsize();

    printf(1,"The process is using %dB\n",msize);

    free(tmp);

    printf(1,"Free memory\n");

    msize=memsize();

    printf(1,"The process is using %dB\n",msize);

    exit();

}

proc.h

// Per-process state

struct proc {

  uint sz;                     // Size of process memory (bytes)

  pde\_t\* pgdir;                // Page table

  char \*kstack;                // Bottom of kernel stack for this process

  enum procstate state;        // Process state

  int pid;                     // Process ID

  struct proc \*parent;         // Parent process

  struct trapframe \*tf;        // Trap frame for current syscall

  struct context \*context;     // swtch() here to run process

  void \*chan;                  // If non-zero, sleeping on chan

  int killed;                  // If non-zero, have been killed

  struct file \*ofile[NOFILE];  // Open files

  struct inode \*cwd;           // Current directory

  char name[16];               // Process name (debugging)

  int tracemask;

};

proc.c

int

fork(void)

{

  . . .

    // copy trace mask

    np->tracemask = curproc->tracemask;

  . . .

}

ssu\_trace.c

#include "types.h"

#include "stat.h"

#include "user.h"

#include "fcntl.h"

#include "syscall.h"

#include "param.h"

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

    if(argc <= 1){

        exit();

    }

    // if(argc!=3){

    //     printf(1, "input error\n");

    //     exit();

    // }

    //printf(1, "argv[0]: %s,argv[1]: %s,argv[2]: %s, maskLength : %d\n", argv[0], argv[1], argv[2],strlen(argv[1]));

    int mask=0;

    char \*s=argv[1];

    while('0' <= \*s && \*s <= '9')

      mask = mask\*10 + \*s++ - '0';

    //printf(1, "argv[0]: %s,argv[1]: %s,argv[2]: %s, maskLength : %d, mask : %d\n", argv[0], argv[1], argv[2],strlen(argv[1]),mask);

  int i;

  char \*command[MAXARG];

  if (trace(mask) < 0)

  {

    printf(1, "ssu\_trace fail\n");

    exit();

  }

  for (i = 2; i < argc && i < MAXARG; i++)

  {

    command[i - 2] = argv[i];

  }

  exec(command[0],command);

  exit();

}

usys.S

텍스트이(가) 표시된 사진

자동 생성된 설명

Makefile

UPROGS=\

    \_cat\

    \_echo\

    \_forktest\

    \_grep\

    \_init\

    \_kill\

    \_ln\

    \_ls\

    \_mkdir\

    \_rm\

    \_sh\

    \_stressfs\

    \_usertests\

    \_wc\

    \_zombie\

    \_helloworld\

    \_hcat\

    \_ssu\_login\

    \_memsizetest\

    \_ssu\_trace\

EXTRA=\

    mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\

    ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c\

    printf.c umalloc.c helloworld.c hcat.c ssu\_login.c memsizetest.c ssu\_trace.c\

    README list.txt dot-bochsrc \*.pl toc.\* runoff runoff1 runoff.list\

    .gdbinit.tmpl gdbutil\