Universidade do Minho
Escola de Engenharia

Disciplina

Sistemas Operativos

Ano Lectivo de 2013/2014

Grupo 15

José Pereira (a67680), Pedro Cunha (a67677)

Maio/2014

Índice

Resumo

Este relatório serve de apoio ao projecto da disciplina de Sistemas Operativos, desenvolvido pelo grupo 15 composto por: José Pereira (a67680) e Pedro Cunha (a67677)

Foi proposto aos alunos que criassem um software, em linguagem C, aplicando as técnicas e conhecimentos leccionados nas aulas práticas, um programa que consistia numa biblioteca para um cliente usar e num “Servidor”, em que o Cliente pode enviar ao servidor dados sobre um dado acontecimento.

Explicação de Código:

Servidor

```
mkfifo("fifo", 0666);

pipeRead = open("fifo", O_RDONLY);

str[0] = ' ';

while(1)
{
        lixo = read(pipeRead, str, BUFFER_SERV);
        if (lixo <= 0)
        {
                close(pipeRead);
                pipeRead = open("fifo", O_RDONLY);
                continue;
        }

        sscanf(str, "%d", &sizeArg);
        instruct = malloc(sizeof(char *) * sizeArg);
```

O servidor começa por receber do cliente a informação através de um pipe com nome, primeiro recebe o número de strings que ira receber para executar a query pedida. Depois recebe também através do pipe todos os parâmetros da query e envia a função “execArg”.

```
struct listaContagem {
        int cont;
        int myId;
        char campo[128];
        struct listaContagem *lowerLvl;
        struct listaContagem *next;
};
```

O servidor usa esta estrutura de dados, em que guarda o nível superior da contagem. Nos processos filhos esta estrutura também irá guardar os níveis inferiores para ter um tempo de resposta superior nas queries de agregar.

```c
int execArg(char *instruct[], int sizeArg)
{
        int fpid = 0, fpipe, logfile, i = 0, j = 0, lixo, status = 15;
        char rndstr[BUFFER_SERV];
        struct listaContagem *apontaTopo = NULL;

        apontaTopo = getTopLvl(instruct[0]);

        if(apontaTopo->myId == 0)
        {
                mkfifo(instruct[0], 0666);

                fpid = fork();
                if(fpid == 0)
                {
                        topLvlProcess((*apontaTopo), 0, 0);
                        exit(0);
                }

                apontaTopo->myId = fpid;
        }
```

Está função, com ajuda da função "getTopLvl", aloca espaço ou encontra a estrutura que se refere a query em questão, depois caso seja preciso abre um processo para o mesmo, ela comunica através de um pipe com nome para o processo criado. Esta função trata também dos logs, guardando num ficheiro e verificando a última query executada. Ela verifica com "waitpid(apontaTopo->myId, &status, WNOHANG)" se a função, caso já exista, se ainda está a correr, caso não esteja reinicia a função e repõe os dados.

```c
fpipe = open(myList->campo, O_RDONLY);

while(1)
{
        n = read(fpipe, str, BUFFER_SERV);
        if (n <= 0)
        {
                close(fpipe);
                fpipe = open(myList->campo, O_RDONLY);
                continue;
        }

        sscanf(str, "%d", &sizeArg); sizeArg--;
        comando = malloc(sizeof(char *) * sizeArg);
```

Nos processos dos níveis superiores começam por receber a query e depois enviar para a função respectiva, "incrementFunc" caso seja para incrementar um contador ou "agregarFunc" para agregar. E para notar que a função de incremento de contadores guarda também na estrutura de dados para depois a função de agregar ter um tempo de resposta melhor.

Conclusão

Após a resolução de todo o projecto, o grupo concluiu que o projecto foi deveras aliciante e pôs em prática todos os conhecimentos e técnicas aprendidas ao decorrer do semestre em Sistemas Operativos.

Todos os elementos se empenharam e sempre deram o seu melhor no decorrer do trabalho ajudando sempre com os conhecimentos que possuíam.

Finalmente, concluímos, que apesar de dúvidas, houve poucos contratempos face à resolução de todo o nosso projecto.