DEPARTAMENTO DE ENGENHARIA INFORMÁTICA E DE SISTEMAS
INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

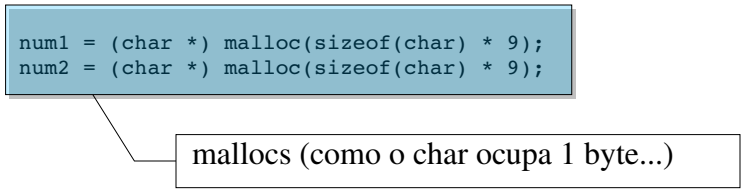
Sistemas Operativos I
2003/2004

Relatório do trabalho prático

21120274 – André Lemos (a_lemos@alunos.deis.isec.pt)

O trabalho começou pela análise da função getopt() que parecia complicada de utilizar de início, mas tornou-se razoavelmente fácil devido à quantidade de exemplos que se encontram no mundo do *opensource*, eu em particular, recorri ao código de fonte de alguns utilitários do sistema operativo *OpenBSD*, recorrendo também a alguma documentação encontrada na internet.

Na gama dos números fornecidos pelos utilizadores, optei por usar um long int para armazenar os valores da gama fornecidos pelo utilizador, e como tal tive que usar a função atol(), em deterimento da função atoi() para que pudessem ser usados números com 9 digitos. Após alguma utilização do programa, e especialmente depois de experimentar o compilador da intel (icc), deparei-me com o problema de que por vezes o programa terminava de forma anormal, por vezes no gcc, mas sempre no icc. Para resolver o problema tive que alocar memória para estes dois números para que nunca houvesse problema.



Para além deste tratamento dos dados iniciais, foram também impostas algumas condições para se ter a certeza de que o utilizador não se enganava no input das várias variáveis na linha de comandos.

Após a verificação o programa divide a gama fornecida pelo utilizador pelos vários processos, se tiver sido especificado o número de processos desejados pelo utilizador, ou calcula o número de processos necessários para satisfazer a condição do utilizador para um determinado tamanho de uma sub-intervalo. Como a divisão pode nem sempre ser certa, teve que se ter o cuidado de adicionar o resto da divisão à última gama que iria ser calculada, para assim cobrir completamente a gama fornecida pelo utilizador.

A invocação do pai() recebe como parâmetros, o pipe, o número de primos a calcular (ou fornecidos pelo utilizador, ou no caso de não terem sido fornecidos, os que estão estabelecidos em `getopt.h`), nome do ficheiro de log, ponteiro para o ficheiro já aberto pelo main() e número de filhos que foram criados.

A função pai() tem como função ler do pipe os valores armazenados pelos filhos até que o número de primos especificado seja atingido, enquanto este valor não é atingido, os números primos, e os processos que os encontraram vão sendo mostrados no ecrã, ou escritos num ficheiro, conforme a opção do utilizador. Depois de serem encontrados todos os primos desejados, são terminados todos os filhos que ainda estiverem a fazer cálculos, e o utilizador é informado disso, caso não tenha optado por gravar os primos num log.

Para fazer este trabalho, foi necessário primeiro relembrar o conceito de número primo para que se pudesse fazer uma função que os calculasse.

Ora um número primo, não é nada mais do que um número que não é completamente divisivel por mais nenhum número, excepto por si próprio, e pela unidade. Visto isto, podemos passar à análise da função que calculava os primos.

Esta função encontra-se no ficheiro `calc.c` por motivos de arrumação do código, separando assim a parte do cálculo e processo filho do resto do código.

Como na descrição do trabalho está que os filhos deveriam comunicar ao processo pai tanto a sua identifiacação (pid), como os números primos que iriam calculando, recorri ao uso de uma estrutura que iria passar ao processo pai estes valores.

```
typedef struct {
     pid_t processo;
     int numero;
} recebe;
```

estrutura para passar o pid e o primo para o pai

Esta função recebe do processo pai 2 valores, um valor superior de uma gama, e um valor inferior. Esta função vai calcular todos os primos entre estes dois valores.

Para calcular os primos, bastou criar um ciclo onde para cada número dentro do intervalo seria calculado o resto da divisão de todos os números inferiores a ele, à excepção de 0 e de 1. Se o resto da divisão fosse 0, então é porque o número a ser testado, não era primo.

Se esta condição nunca acontecesse, então o número era armazenado num pipe (`pdes`) para o processo pai posteriormente o poder ler.

Um dos problemas encontrados no uso de pipes, foi o de que se o pipe estivesse fazio, o programa parava na função read() à espera de mais lados para ler, para ultrapassar este problema, poder-se-ia ter utilizado o O_NONBLOCK que evitaria que o problema, mas seria pouco útil porque não permitiria destinguir qual era o processo que tinha calculado que gama. Para resolver este problema, no fim de cada filho ter processado a sua gama, é inserido no pipe o valor de um primo como sendo -1, podendo assim o processo pai verificar que este processo havia terminado os seus cálculos, visto o cálculo da gama ser apenas sobre números positivos, não havia problemas de acidentes.

Referências:

Para a realização deste trabalho, foi consultada a seguinte documentação:

http://www.opengroup.org/onlinepubs/009695399/functions/getopt.html
http://www.gnu.org/software/libc/manual/html_node/Getopt.html
http://www.numaboa.com.br/criptologia/matematica/primos.php
apontamentos das aulas práticas, nomeadamente da ficha nº 6, 7 e 8