SIMULADOR DE CAMBIO DE CONTEXTO

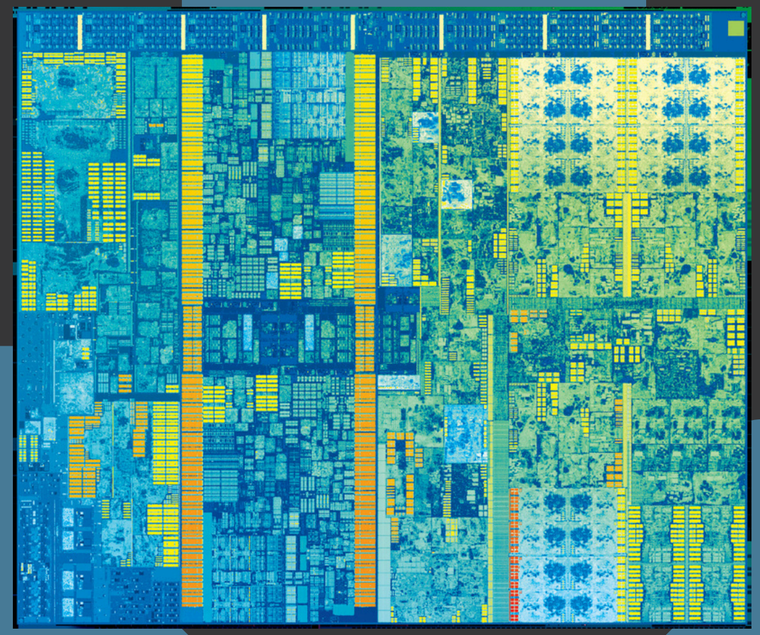
INFORME FINAL: PROYECTO 1

DAVID QUINTERO G.
ANDRÈS ECHEVERRI B.
VIVIANA ARANGO T.

Tabla de Contenido

Objetivo del proyecto

El objetivo fue implementar un simulador de cambio de contexto entre procesos, usando el lenguaje C bajo Linux, para comprender cómo el sistema operativo guarda y recupera el estado de los procesos cuando cambia cuál está usando el CPU, siguiendo una política de planificación Round-Robin.



Descripcion del funcionamiento

Descripcion del funcionamiento

Entrada del sistema

El programa recibe el archivo procesos.txt con este formato:

PID:1,AX=2,BX=3,CX=1,Quantum=3
PID:2,AX=4,BX=2,CX=0,Quantum=2

Estructura del proceso

```
typedef struct {
    int pid;
    int pc;
    int ax, bx, cx;
    int quantum;
    char estado[10]; //
"Listo", "Ejecutando",
"Terminado"
} Proceso;
```

Planificacion Round-Robin

Cada proceso ejecuta hasta agotar su quantum. Si no termina, pasa al final de la cola.
Se pueden generar interrupciones, lo que bloquea temporalmente el proceso

Descripcion del funcionamiento

Instrucciones implementadas

- ADD AX, BX
- INC AX
- SUB CX, 1
- MUL AX, 2
- SUB AX, BX
- INC CX
- NOP

Salida

La simulación genera un archivo simulacion.log que muestra:

- Cambios de contexto
- Estados de procesos
- Interrupciones simuladas

Salida

- Desbloqueos de procesos
- Finalización de procesos

Ejecucion del programa en terminal

EJECUCION DEL PROGRAMA

01

Compilacion

gcc -fexec-charset=UTF-8 main.c -o procplanner

02

Ejecucion

./procplanner -f procesos.txt

03

Resultado

Se genera el archivo simulacion.log

04

Estado final

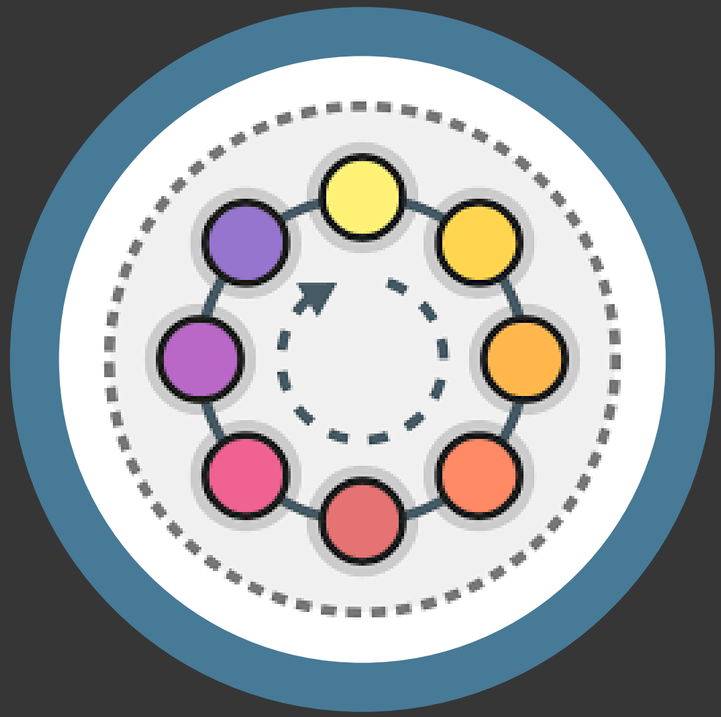
Proceso 1 terminado.
Proceso 2 terminado.

ARCHIVO SIMULACION.LOG EJEMPLO

03

[Cambio de contexto]
Cargando estado del Proceso 1
 -> Ejecutando: ADD AX,BX | Resultado: AX=5
 -> Ejecutando: INC AX | Resultado: AX=6
 -> Ejecutando: SUB CX,1 | Resultado: CX=0

[Cambio de contexto]
Cargando estado del Proceso 2
 -> Ejecutando: MUL AX,2 | Resultado: AX=8
!! Interrupción simulada en Proceso 2. Se bloquea antes de agotar quantum.
...

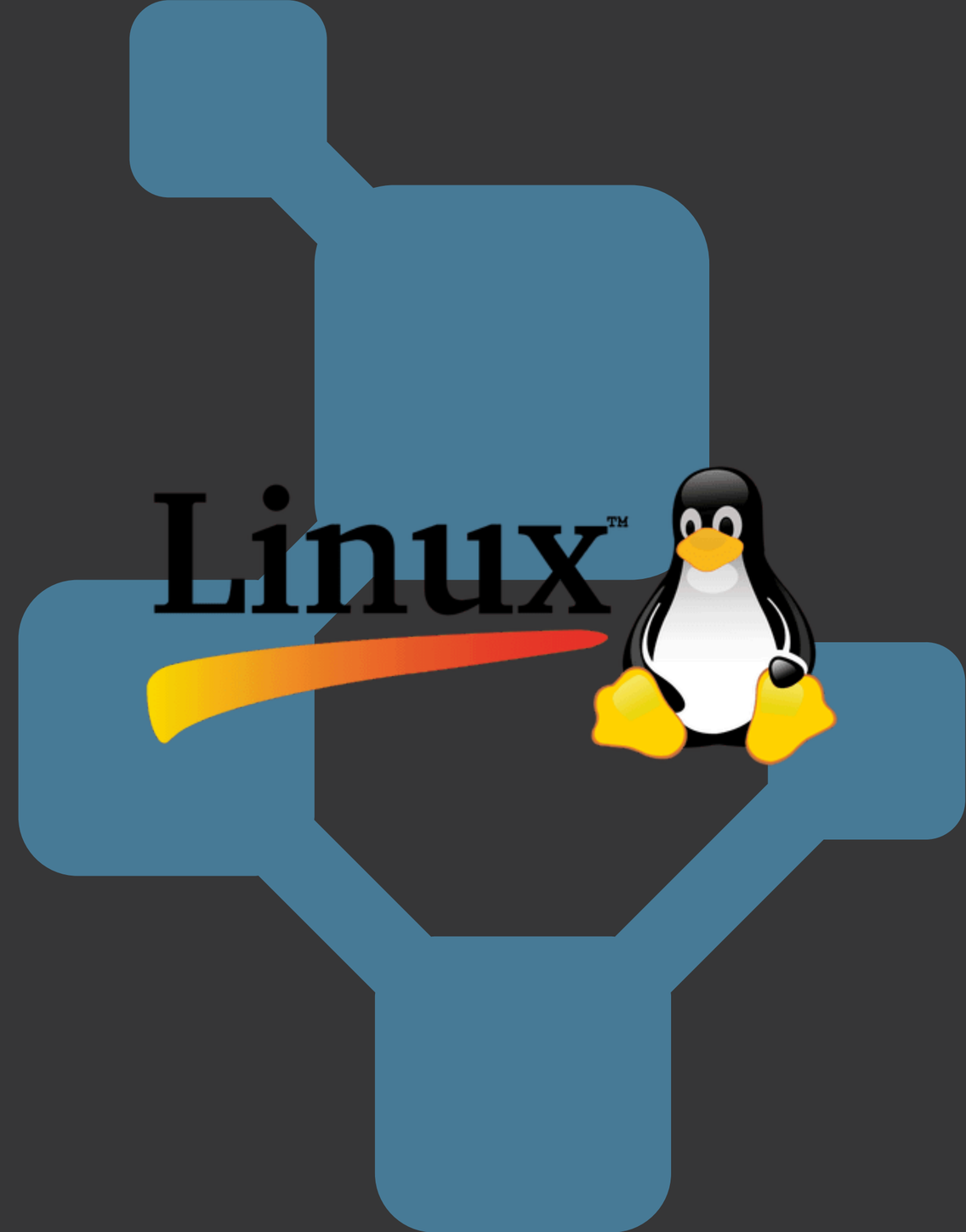


Descripcion del hardware

- **Arquitectura:** x86_64
- **CPU OP-MODE(S):** 32-bit 64-bit
- **Model Name:** Intel(R) Core(TM) i7-8550U CPU @ 1.80GH
- **Maxima RAM utilizada:** 976 kbytes
- **Compilador usado:** gcc (Ubuntu 13.2.0)
- **RAM total del sistema:** 15 GB

TABLA DE EJECUCION

PROCESO	QUANTUM	INSTRUCCIONES EJECUTADAS	RESULTADO FINAL
1	3	ADD AX,BX, INC AX, SUB CX,1, NOP	AX=6, CX=0
2	2	MUL AX,2, SUB AX,BX, INC CX, NOP	AX=6, CX=1



Conclusiones

Conclusiones

01

Uso extremadamente bajo de RAM El programa solo consumió 976 KB, lo cual representa aproximadamente un 0,0065 % de los 15 GB disponibles. Esto confirma que el simulador es altamente eficiente en consumo de memoria.

02

Procesador subutilizado El simulador corrió en un Intel Core i7-8550U, pero no usa más de un hilo ni tareas paralelas. Esto significa que incluso procesadores de gama baja pueden ejecutarlo sin problemas.

03

Escalabilidad posible Dado el bajísimo consumo de recursos, se podría: Aumentar el número de procesos Añadir más instrucciones por proceso Extender la lógica (más instrucciones, más ciclos, simulación en tiempo real) Sin comprometer estabilidad ni rendimiento.

Gracias

:)