Relatório do Simulador de Gestão de Processos

ID do grupo: 21
Elementos do grupo:
- **41059 - Rúben Guilherme**
- **41853 - Paulo Duarte**
- **41872 - Rui Barata**

GIT: https://gitlab.com/the-releasers/so-project

Índice

Introdução

Neste projeto pretende-se desenvolver um simulador de gestão e execução de processos que tem por base os conhecimentos adquiridos na unidade curricular de Sistemas Operativos na Universidade da Beira Interior.

Com este projeto para além de ser pretendido o aprofundamento dos principais temas da unidade curricular em questão (como por exemplo: os algoritmos de escalonamento de processos, de gestão de memória e a forma como o processador os gere), também é importante o desenvolvimento das nossas capacidades de programação num projeto com uma elevada dimensão e complexidade.

Funcionamento do Simulador

O simulador de gestão de processos desenvolvido pela equipa de programadores tem diversas funcionalidades, tais como:

- vários algoritmos de escalonamento de processos com e sem preempção;
- vários algoritmos de gestão de memória;
- dois modos de utilização: modo debug e modo de execução normal;
- a possibilidade de alterar o time quantum do simulador.

A diferença entre o modo de execução normal e o modo de debug é que o modo de execução normal deverá executar as instruções do gestor de processos a partir do ficheiro plan.txt, ao contrário de que no modo de debug a execução das instruções do gestor de processos deverão ser efetuadas através do input em tempo-real por parte do utilizador.

Para iniciar o simulador:

1. Fazer *make* na shell.
2. Iniciar o simulador inserindo *./Simulador* para o modo de execução normal ou *./Simulador -d* para o modo de debug

Exemplo de execução

Configuração dos ficheiros:

```
C:\Users\ruben\Desktop\UC\Segundo Ano\Segundo Semestre\SO\Projeto\teste\so-project>type plan.txt
progenitor.prg 0 1
progenitor.prg 20 1
filho1.prg 30 1
```

```
C:\Users\ruben\Desktop\UC\Segundo Ano\Segundo Semestre\SO\Projeto\teste\so-project>type control.txt
E
R
E
R
E
R
T
```

```
C:\Users\ruben\Desktop\UC\Segundo Ano\Segundo Semestre\SO\Projeto\teste\so-project>type progenitor.prg
M 100
H
H
A 19
A 20
S 12
A 1
A 4
C 2
L filho1
C 2
L filho2
T
```

```
C:\Users\ruben\Desktop\UC\Segundo Ano\Segundo Semestre\SO\Projeto\teste\so-project>type filho1.prg
H
M 200
A 19
T
```

```
C:\Users\ruben\Desktop\UC\Segundo Ano\Segundo Semestre\SO\Projeto\teste\so-project>type filho2.prg
H
M 300
S 12
A 5
T
```

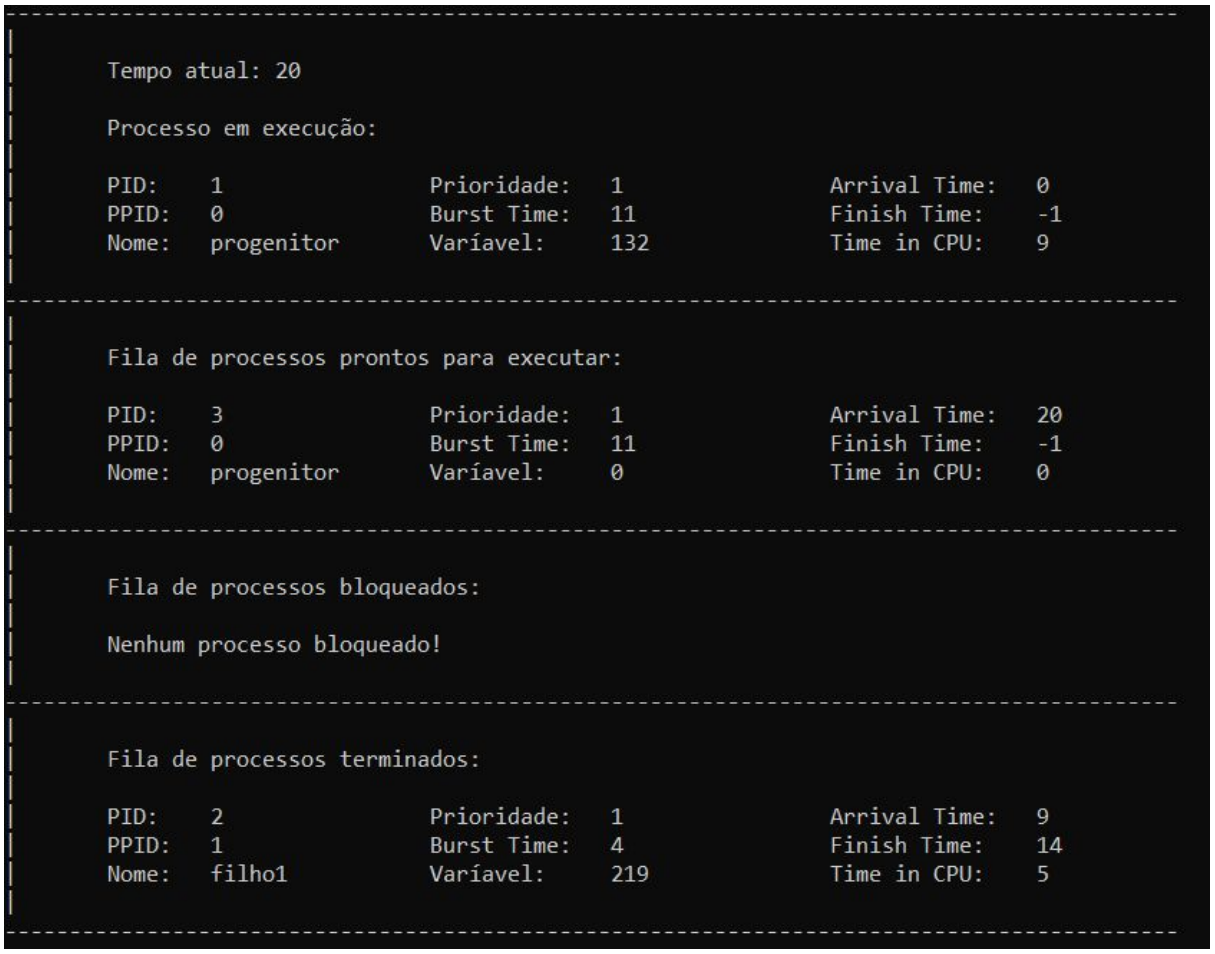
Menu :

```
------------------------------------------------------------------------------------------
|
|       Escalonamento Selecionado:   First Come First Serve
|       Tipo de Escalonamento:       Não preemptivo
|       Time Quantum:                10
|       Modo de Debug:               Desativado
|       Número de partições:         128
|
|
|       Menu
|
|       1 - Iniciar
|       2 - Opções
|
|       0 - Sair
|
------------------------------------------------------------------------------------------
>> 1
```

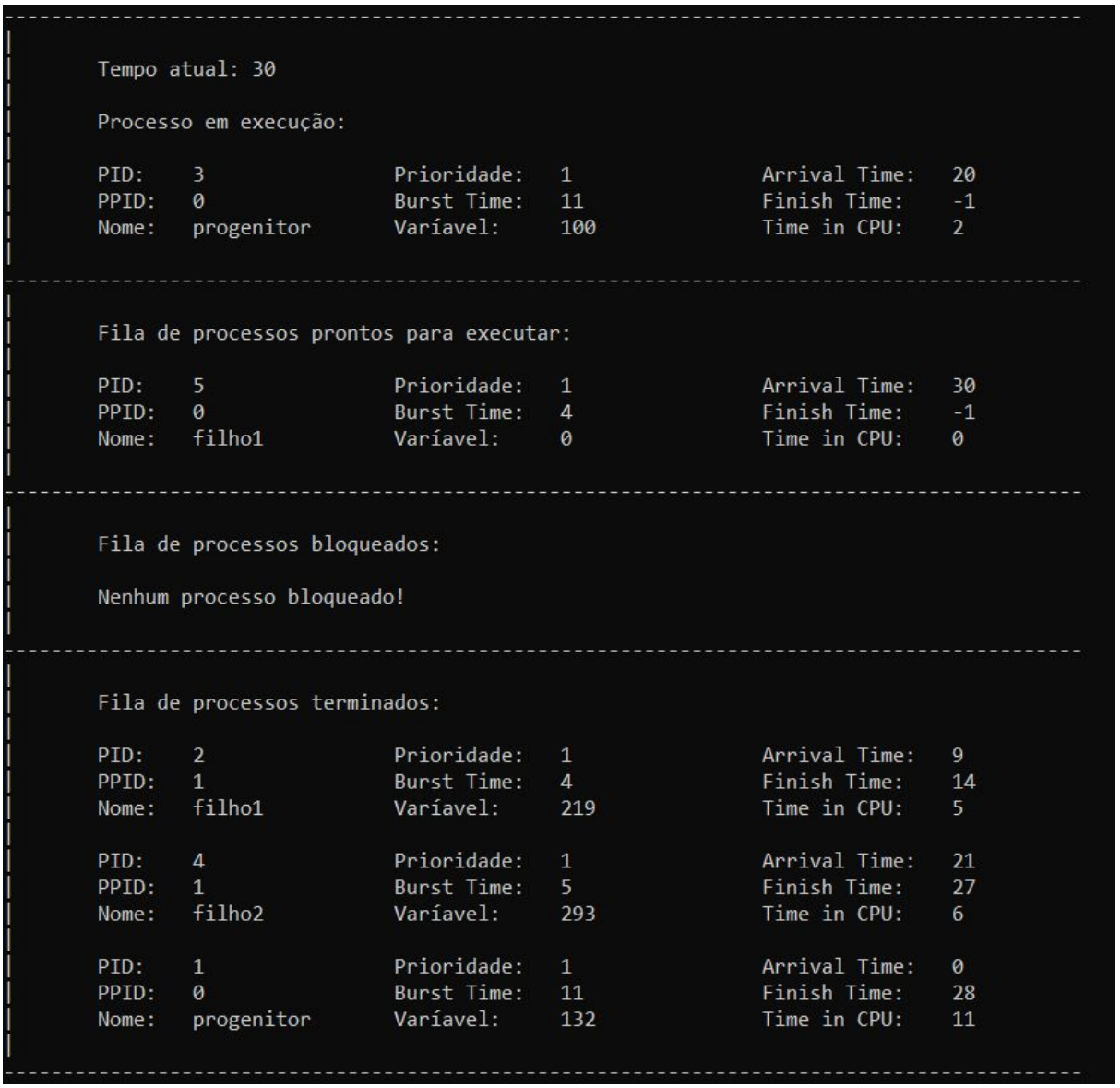
Primeiro Report :

```
------------------------------------------------------------------------------------------
|
|       Tempo atual: 10
|
|       Processo em execução:
|
|       PID:    2               Prioridade:   1               Arrival Time:   9
|       PPID:   1               Burst Time:   4               Finish Time:    -1
|       Nome:   filho1          Varíavel:     0               Time in CPU:    1
|
------------------------------------------------------------------------------------------
|
|       Fila de processos prontos para executar:
|
|       Nenhum processo pronto para executar!
|
------------------------------------------------------------------------------------------
|
|       Fila de processos bloqueados:
|
|       Nenhum processo bloqueado!
|
------------------------------------------------------------------------------------------
|
|       Fila de processos terminados:
|
|       Nenhum processo terminado!
|
------------------------------------------------------------------------------------------
```

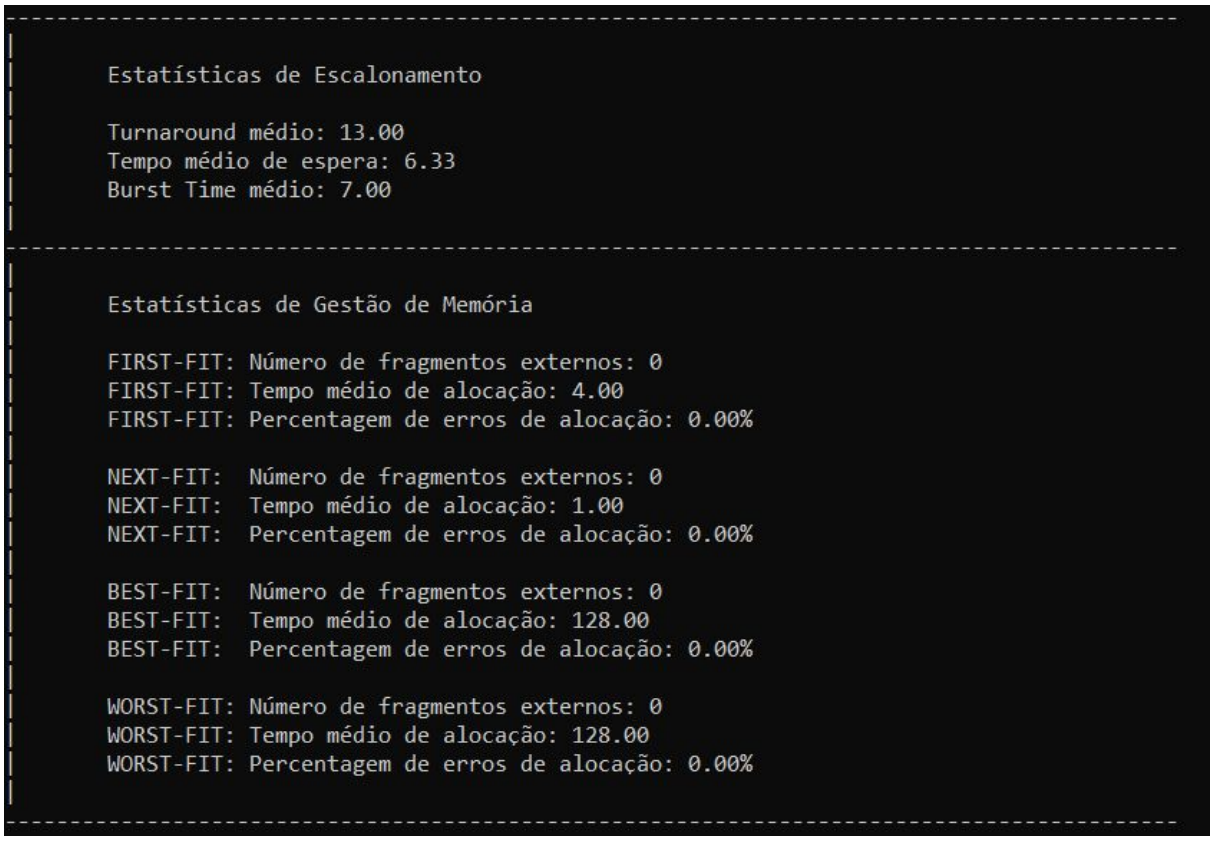
Segundo Report:



Terceiro Report:



Estatísticas Finais



Participações no desenvolvimento do Simulador

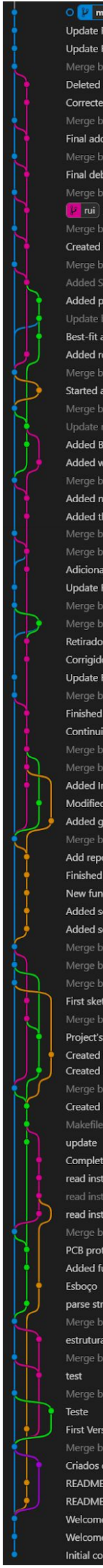
Na página seguinte estão colocadas algumas images que representam as participações que cada elemento teve ao longo do projeto juntamente com a data da participação:

Notas:

Usernames de cada elemento:

- rubenhg - Rúben Guilherme
- p_duarte - Paulo Duarte
- Rui00Barata - Rui Barata

Alguns commits foram feitos em conjunto durante uma reunião, estes estão marcados com "**Done during a Meeting**".



master Update README.md	14 Jun 2020 12:55	Rúben Guilherme
Update README.md	14 Jun 2020 12:50	Rúben Guilherme
Update README.md	14 Jun 2020 12:50	Rúben Guilherme
Merge branch 'rubenhg' into 'master'	14 Jun 2020 12:48	Rúben Guilherme
Deleted unnecessary comments.	14 Jun 2020 12:11	rubenhg
Corrected an error in the SJF function	14 Jun 2020 11:49	rubenhg
Merge branch 'rubenhg' into 'master'	13 Jun 2020 20:05	Rúben Guilherme
Final additions to the simulator. Changed the program's design. Done during a meeting	13 Jun 2020 20:04	rubenhg
Merge branch 'rubenhg' into 'master'	13 Jun 2020 16:03	Rúben Guilherme
Final debugging. Done during a meeting.	13 Jun 2020 16:02	rubenhg
Merge branch 'rubenhg' into 'master'	12 Jun 2020 15:39	Rúben Guilherme
rui Minor bugfixes	12 Jun 2020 15:38	rubenhg
Merge branch 'rubenhg' into 'master'	12 Jun 2020 13:41	Rúben Guilherme
Created the report for memory management. Started testing the new schedulling algorithms and the memory allocation/deallocation algorithms. Some minor bugfixes. Done during a Meeting	12 Jun 2020 13:41	rubenhg
Merge branch 'rubenhg' into 'master'	11 Jun 2020 20:58	Rúben Guilherme
Added Shortest Job First Schedulling with and without preemption.	11 Jun 2020 20:24	rubenhg
Added priority scheduling without preemption	11 Jun 2020 18:35	p_duarte
Update best-fit algorithm	11 Jun 2020 17:31	p_duarte
Best-fit algorithm concluded	11 Jun 2020 17:10	p_duarte
Added round-robin algorithm for short schedulling	11 Jun 2020 17:09	Rui00Barata
Merge branch 'rubenhg' into 'master'	11 Jun 2020 16:13	Rúben Guilherme
Started adding preemption. Added Priority Sched. w/Preemption. Done during a meeting	11 Jun 2020 16:12	rubenhg
Merge branch 'paulo' into 'master'	11 Jun 2020 14:25	Rúben Guilherme
Update memory.ml	11 Jun 2020 14:23	p_duarte
Added Best-Fit Algorithm	11 Jun 2020 12:54	p_duarte
Added worst-fit allocation	11 Jun 2020 11:49	Rui00Barata
Merge branch 'rubenhg' into 'master'	10 Jun 2020 19:53	Rúben Guilherme
Added next-fit allocation	10 Jun 2020 19:04	rubenhg
Added the memory module with the first fit allocation. Done during a meeting.	10 Jun 2020 15:16	rubenhg
Merge branch 'rubenhg' into 'master'	15 May 2020 16:54	Rúben Guilherme
Merge branch 'master' of gitlab.com:the-releasers/so-project into rubenhg	15 May 2020 16:53	rubenhg
Adicionado o resumo do estado do projeto ao relatório	15 May 2020 16:53	rubenhg
Update README.md	15 May 2020 15:36	Rúben Guilherme
Merge branch 'rubenhg' into 'master'	15 May 2020 15:36	Rúben Guilherme
Merge branch 'master' into 'rubenhg'	15 May 2020 15:35	Rúben Guilherme
Retirados alguns printf's usados para debug	15 May 2020 15:32	rubenhg
Corrigidos alguns erros e adicionado o relatório	15 May 2020 14:40	rubenhg
Update README.md	12 May 2020 11:14	Rúben Guilherme
Merge branch 'rubenhg' into 'master'	12 May 2020 11:13	Rúben Guilherme
Finished the first part of the project. Done during a group meeting.	12 May 2020 11:11	rubenhg
Continuing debugging during a meeting.	9 May 2020 17:46	Rui00Barata
Merge branch 'rubenhg' of gitlab.com:the-releasers/so-project into rui	9 May 2020 16:11	Rui00Barata
Merge branch 'paulo' of gitlab.com:the-releasers/so-project into rui	9 May 2020 16:08	Rui00Barata
Added Interrupt function.	9 May 2020 16:07	Rui00Barata
Modified menu to be more legible.	9 May 2020 16:04	rubenhg
Added global report function.	9 May 2020 15:57	p_duarte
Merge branch 'paulo' into 'master'	9 May 2020 12:49	Paulo Duarte
Add report function. Continued debugging the simulator. Done during a meeting	9 May 2020 12:47	p_duarte
Finished developing major features. Started debugging the simulator. Done during a meeting.	8 May 2020 19:17	rubenhg
New function to choose the short-term scheduller	7 May 2020 17:48	Rui00Barata
Added some features and corrections during a group meeting.	7 May 2020 15:46	Rui00Barata
Added some features and corrections during a group meeting.	7 May 2020 15:44	Rui00Barata
Merge branch 'rui' into 'master'	7 May 2020 14:10	Rui Pedro de Al...
Merge branch 'paulo' into 'master'	7 May 2020 14:09	Paulo Duarte
Merge branch 'rubenhg' into 'master'	7 May 2020 14:05	Rúben Guilherme
First sketch of FCFS algorithm and necessary functions	6 May 2020 18:50	Rui00Barata
Merge branch 'paulo' of gitlab.com:the-releasers/so-project into rui	6 May 2020 15:48	Rui00Barata
Project's menu and integration (in development)	6 May 2020 12:59	p_duarte
Created long-term scheduller.	6 May 2020 12:31	rubenhg
Created execute function. Created in meeting with all members.	2 May 2020 16:24	p_duarte
Merge branch 'rubenhg' into 'master'	1 May 2020 17:14	Rúben Guilherme
Created functions for the process manager. Modified some types on the Lib Module. Made during a meeting.	1 May 2020 17:10	rubenhg
Makefile created. Additional files created.	1 May 2020 14:55	rubenhg
update	1 May 2020 10:55	rubenhg
Completed read_instr	1 May 2020 10:22	Rui00Barata
read instruction	30 Apr 2020 10:57	p_duarte
read instruction	30 Apr 2020 10:55	p_duarte
read instruction	30 Apr 2020 10:51	p_duarte
Merge branch 'rubenhg' into 'master'	15 Apr 2020 12:45	Rúben Guilherme
PCB prototype	15 Apr 2020 12:40	rubenhg
Added func: abrir, tirar_CR; Added var: memory_model, ultimo;	15 Apr 2020 11:41	rubenhg
Esboço	13 Apr 2020 18:11	p_duarte
parse string to instruction type	11 Apr 2020 15:52	rubenhg
Merge branch 'rui' into 'master'	11 Apr 2020 13:24	Rúben Guilherme
estruturas feitas	9 Apr 2020 14:48	Rui00Barata
Merge branch 'rui' into 'master'	8 Apr 2020 17:30	Rui Pedro de Al...
test	8 Apr 2020 17:26	Rui00Barata
Merge branch 'rubenhg' into 'master'	8 Apr 2020 17:22	Rúben Guilherme
Teste	8 Apr 2020 17:20	rubenhg
First Version	8 Apr 2020 16:50	p_duarte
Merge branch 'rubenhg' into 'master'	8 Apr 2020 16:43	Paulo Duarte
Criados os primeiros modulos	8 Apr 2020 16:37	rubenhg
README	8 Apr 2020 16:36	p_duarte
README	8 Apr 2020 16:34	p_duarte
Welcome Message	7 Apr 2020 19:11	p_duarte
Welcome Message	7 Apr 2020 18:09	Paulo Duarte
Initial commit	7 Apr 2020 18:08	Paulo Duarte

Algoritmos

Escalonamento Não Preemptivo

FCFS (First Come First Serve)

Este algoritmo não preemptivo só atua quando um processo acaba.

Logo que um processo termine um novo processo é retirado logo do início da readyQ (Fila com os processos prontos a executar) pois este foi o primeiro que entrou na fila e será o primeiro a sair

(Se o processo não for encontrado na pcb_table(tabela com todos os processos), então a função findProcInd devolve -1 e então é imprimido no ecrã um erro).

Mas se o ind for >= 0 vamos alterar a nossa variável que guarda as informações do processo que está a executar.

SJF (Shortest Job First)

O algoritmo de Shortest Job First funciona de uma forma muito parecida ao algoritmo de escalonamento por prioridade.

Quando este algoritmo é chamado começa por comparar os processos na readyQ através de um algoritmo auxiliar que percorre esta fila e devolve aquele com o shortest job (calculado usando: burst time - pc).

Depois vamos alterar a nossa variável que guarda as informações do processo que está a executar com os dados do processo retornado pela função auxiliar (Para obter o índice na pcb_table recorremos à função findProcInd que neste algoritmo não dá erro visto estar assegurado que o processo em questão existe (pois estava na readyQ).

Prioridade

O algoritmo de prioridade sem preempção quando é chamado verifica se existe algum processo em execução. Se não existir nenhum processo em execução chamamos um algoritmo auxiliar que percorre a readyQ para devolver o processo com maior prioridade (maior prioridade ⇔ valor mais baixo).

De seguida, alteramos a nossa variável que guarda as informações do processo que está a executar com as informações do processo que está a executar. (Conseguimos obter o índice do processo através de uma função auxiliar (findProcInd) que devolve o índice do processo pretendido).

Escalonamento Preemptivo

Round Robin

O round robin só atua quando um processo ou o tempo restante do processo que está a executar(dado pelo rem_time) acaba.

Logo que isto acontece um novo processo é retirado logo do início da readyQ pois este foi o primeiro que entrou na fila e será o primeiro a ser executado a seguir.

(Se o processo não for encontrado na pcb_table, então a função findProcInd devolve -1 e então é imprimido no ecrã um erro).

Mas se o ind for >= 0 vamos alterar a nossa variável que guarda as informações do processo que está a executar.

SJF (Shortest Job First)

O algoritmo de Shortest Job First funciona de uma forma muito parecida ao algoritmo de escalonamento por prioridade.

Quando este algoritmo é chamado começa por verificar se já estava anteriormente algum processo em execução. Se não estiver vai executar o algoritmo como se fosse não preemptivo. Se tivermos algum processo guardado na estrutura de dados usada para guardar os dados do processo a executar então vamos executar o algoritmo auxiliar que vai ser apresentado com um parâmetro diferente, usando o (burst time - pc) do processo que estava guardado (Pois este não está na readyQ e queremos que seja usado na comparação. O algoritmo auxiliar percorrer a readyQ e devolver o processo com o shortest job (calculado usando: burst time - pc).

Depois vamos alterar a nossa variável que guarda as informações do processo que está a executar com os dados do processo retornado pela função auxiliar (Para obter o índice na pcb_table recorremos à função findProcInd que neste algoritmo não dá erro visto estar assegurado que o processo em questão existe (pois estava na readyQ).

Prioridade

O algoritmo de prioridade com preempção começa com a verificação se existe algum processo em execução. Se não existir nenhum processo em execução, então o algoritmo vai executar como se fosse não preemptivo. Se existir algum processo em execução, então vamos chamar uma função auxiliar para devolver o processo com maior prioridade da readyQ. Se o processo devolvido for o mesmo que o processo que está em execução, então o algoritmo não faz nada, visto que o processo que está a executar é o processo com maior prioridade. Caso o processo devolvido seja diferente do processo em execução, então o algoritmo vai mover o processo em execução para a readyQ e colocar o processo com maior prioridade em execução alterando os campos da variável que guarda as informações do processo que está a executar.

Escalonamento a longo prazo

O algoritmo de escalonamento a longo serve para desbloquear processos. O algoritmo vai percorrer a fila de processos bloqueados e selecionar no mínimo um para ser desbloqueado (todos os processos têm uma chance de 1/N de serem desbloqueados, onde N é o número de processos da blockedQ (Fila de processos bloqueados).

Depois de selecionados os processos que vão ser desbloqueados, recorre-se a um algoritmo auxiliar que percorre a blockedQ e retira de lá os que foram selecionados e insere-os na readyQ.

Alocação de memória

FIRST FIT

O algoritmo de First Fit vai selecionar o primeiro endereço da memória dinâmica onde haja memória suficiente para as alocações que são precisas alocar. Se encontrar este índice, vai substituir na memória com o valor do PID do processo que precisou da alocação. (Para saber se há memória suficiente é utilizado um algoritmo que recebe um índice e retorna true se houver espaço contíguo suficiente). Se não houver um indíce que satisfaça estas condições então vai devolver (-1).

NEXT FIT

O algoritmo de Next Fit vai selecionar o primeiro endereço da memória dinâmica onde haja memória suficiente para as alocações que são precisas alocar. Este algoritmo difere do de First Fit pois não começa a procurar este índice a partir do início da memória mas sim a partir do índice da última alocação. (A primeira alocação funciona como o First Fit). Se encontrar este índice, vai substituir na memória com o valor do PID do processo que precisou da alocação. (Para saber se há memória suficiente é utilizado um algoritmo que recebe um índice e retorna true se houver espaço contíguo suficiente). Se não houver um indíce que satisfaça estas condições então vai devolver (-1).

BEST FIT

O algoritmo de Best Fit vai percorrer a memória dinâmica em busca do endereço com o espaço mínimo suficiente para as alocações necessárias. Se encontrar este espaço, então vai substituir na memória com o valor do PID do processo que solicitou a alocação.

Em primeiro lugar, o algoritmo vai percorrer o array todo para calcular o menor intervalo contíguo de memória disponível que vai ser atribuído a min e a primeira posição deste intervalo estará em ind.

Encontrado o espaço disponível para alocar o número de unidades de memória do processo, o algoritmo irá proceder à substituição de todas as posições de memória solicitadas pelo PID do processo.

No final, se a memória foi alocada então devolvemos o número de nodos percorridos (tamanho do array), se não devolvemos -1.

WORST FIT

O algoritmo Worst Fit tem que ter 4 variáveis inteiras. Duas para guardar o tamanho do maior espaço até ao momento(max) e o index em que começa(maxIndex), e mais duas para guardar o tamanho do espaço que a função irá estar a contar(count) e o seu index(countIndex). Este ainda vai ter um boolean para saber se a alocação de memória foi bem sucedida(flag).

O algoritmo vai percorrer o array todo uma primeira vez para calcular o maior intervalo contíguo de memória disponível que vai ser atribuído a max e a primeira posição deste intervalo estará em maxIndex.

A seguir vamos ver se este intervalo tem memória suficiente para alocar o número de unidades de memória que estão a ser solicitadas com a função has_memory_available(este recebe um índice e retorna true se houver espaço contínuo suficiente). Se recebermos a aprovação da função anterior então vamos preencher todas as posições desde o maxIndex até ao (maxIndex+número de unidades de

memória que estão sendo solicitadas) com o PID do processo que fez a solicitação e alterar o valor da flag para true, assim sabemos que foram alocados processos.

No final se a memória foi alocada então devolvemos o número de nodos percorridos (tamanho do array), se não devolvemos -1.

Desalocação de memória

Este algoritmo vai receber o PID do processo para o qual queremos desalocar a memória e vai percorrer a memória dinâmica desalocando a memória deste processo.

Cálculo dos furos na memória

Este algoritmo vai percorrer a memória dinâmica e calcular o número de furos (fragmentos de tamanho 1 ou 2) na memória. Faz isto percorrendo a memória e utilizando um contador que conta o tamanho dos fragmentos. Se quando chegar a uma parte da memória com memória alocada ou ao fim da memória o contador for igual a 1 ou 2 vai incrementar um segundo contador que vai ser o valor que este algoritmo vai devolver no fim da sua execução (quando acabar de percorrer a memória dinâmica).

Módulos

Exec

Funções

Nome da função	execute
Inputs	i: int
Descrição	Executa uma instrução do processo em execução.
Outputs	unit ()
Efeito Secundário	No caso de este processo ter sido bloqueado ou terminado, então a função irá adicioná-lo à Queue dos processos bloqueados ou à Queue dos processos terminados, conforme o seu estado.

Report

Funções

Nome da função	print_queue
Inputs	process_queue : pcb Queue
Descrição	Esta função itera sobre a Queue que foi passada como argumento fazendo print dos atributos escolhidos de cada processo.
Outputs	unit ()
Efeito Secundário	Nenhum efeito secundário.

Nome da função	report
Inputs	unit ()
Descrição	Faz o report da simulação mostrando todas as informações pertinentes no momento em que é chamada.
Outputs	unit ()
Efeito Secundário	Nenhum efeito secundário.

Nome da função	turnaround
Inputs	unit ()
Descrição	Calcula o turnaround médio de todos os processos terminados.
Outputs	float
Efeito Secundário	Nenhum efeito secundário.

Nome da função	tme
Inputs	unit ()
Descrição	Calcula o tempo médio de espera de todos os processos terminados.
Outputs	float
Efeito Secundário	Nenhum efeito secundário.

Nome da função	burst_time
Inputs	unit ()
Descrição	Calcula o burst time médio de todos os processos.
Outputs	float
Efeito Secundário	Nenhum efeito secundário.

Nome da função	global_report
Inputs	unit ()
Descrição	Dá print das informações globais à simulação mais pertinentes.
Outputs	unit ()
Efeito Secundário	Nenhum efeito secundário.

Sim

Funções

Nome da função	line_to_instr
Inputs	line : string
Descrição	Converte uma instrução : string numa instrução : instruct.
Outputs	instr
Efeito Secundário	Nenhum.

Nome da função	copy_process
Inputs	p : pcb
Descrição	Faz uma cópia do processo passado pelo argumento.
Outputs	pcb
Efeito Secundário	Nenhum efeito secundário.

Nome da função	find_process_name
Inputs	s : string i : int
Descrição	A função deve ser chamada com i = 0. Percorre a pcb_table e se encontrar um processo com o nome igual a “s” então devolve o indice do processo. Se não encontrar devolve -1.
Outputs	int
Efeito Secundário	Nenhum efeito secundário.

Nome da função	openfile
Inputs	newP : newP
Descrição	Se o processo já existir na tabela de processos, então atualiza os seus atributos e põe o processo na tabela de ready. Se não abre o ficheiro de texto .prg onde se encontram as instruções a executar pelo processo, põe as instruções em memória e insere este processo numa tabela de processos e numa tabela de processos ready.
Outputs	unit ()
Efeito Secundário	A seguir à abertura do ficheiro .prg existe a criação do processo : pcb para que seja possível inserí-lo na tabela de processos.

Nome da função	openfile_string
Inputs	str : string
Descrição	Abre o ficheiro .prg e põe as instruções do processo em questão em memória.
Outputs	unit ()
Efeito Secundário	Nenhum efeito secundário.

Nome da função	read_instr
Inputs	process : pcb
Descrição	Procura a instrução que o processo pretende efetuar e executando-a.
Outputs	unit ()
Efeito Secundário	Dependendo da instrução escolhida, os efeitos secundários irão variar.

Long

Funções

Nome da função	unblock
Inputs	v : bool array n : int
Descrição	Percorre o array v e se encontrar um elemento a true então retira o processo bloqueado com esse indice da blocked queue e passa-o para a ready queue. Existe sempre pelo menos um elemento a true
Outputs	unit ()
Efeito Secundário	Altera a blocked queue, retirando no mínimo um processo e passando-o para a ready queue

Nome da função	long_sched
Inputs	n : int
Descrição	Esta função escolhe de forma aleatória um processo para desbloquear (cada um tem 1/n chances de ser desbloqueado), havendo sempre pelo menos um processo a desbloquear. De seguida chama a função unblock.
Outputs	unit ()
Efeito Secundário	Chama a função unblock

Main

Funções

Nome da função	clock
Inputs	unit ()
Descrição	Um while com uma flag que chama todas as funções que fazem com que o simulador funcione. O clock é chamado no menu principal e apenas termina quando o gestor de processos encontra a instrução T no control.txt ou no stdin (dependendo do modo da simulação).
Outputs	unit ()
Efeito Secundário	Diversos efeitos secundários, visto ser a função principal

Nome da função	menu
Inputs	unit ()
Descrição	Menu com 3 opções: começar a simulação, abrir o menu de opções ou sair do programa. Este menu é a primeira função chamada no princípio da execução do simulador.
Outputs	unit ()
Efeito Secundário	Começar o clock; Sair do programa.

Nome da função	options_menu
Inputs	unit ()
Descrição	Menu com 4 opções: entrar no menu de escolha de algoritmo de escalonamento, alterar o tipo de escalonamento, alterar o time quantum do programa e voltar ao menu principal
Outputs	unit ()
Efeito Secundário	Nenhum efeito secundário.

Nome da função	scheduling_menu
Inputs	unit ()
Descrição	Menu com várias opções que permitem escolher o algoritmo de escalonamento sem preempção a usar.
Outputs	unit ()
Efeito Secundário	Nenhum efeito secundário.

Nome da função	scheduling_menup
Inputs	unit ()
Descrição	Menu com várias opções que permitem escolher o algoritmo de escalonamento com preempção a usar.
Outputs	unit ()
Efeito Secundário	Nenhum efeito secundário.

Nome da função	memory_management
Inputs	unit ()
Descrição	Função onde é especificado o tamanho da memória dinâmica e o tamanho das suas partições.
Outputs	unit ()
Efeito Secundário	Nenhum efeito secundário.

Process

Funções

Nome da função	read_plan
Inputs	unit ()
Descrição	Abre o ficheiro plan.txt e introduz a informação daí retirada na newP - fila de processos novos.
Outputs	unit ()
Efeito Secundário	Nenhum efeito secundário.

Nome da função	interrupt
Inputs	unit ()
Descrição	Bloqueia o processo em execução
Outputs	unit ()
Efeito Secundário	Nenhum efeito secundário.

Nome da função	read_command
Inputs	c : char
Descrição	Interpreta o comando recebido (c). 'E' - Executa um processo durante o time_quantum; 'I' - Interrompe o processo em execução; 'D' - Faz o escalonamento a longo prazo, desbloqueando processos; 'R' - Faz um report do estado atual do simulador 'T' - Termina o simulador, colocando a clock_flag a false e faz o report das estatísticas globais
Outputs	unit ()
Efeito Secundário	Depende do comando recebido

Nome da função	read_terminal
Inputs	unit ()
Descrição	Lê uma instrução do stdin e chama o read_command
Outputs	unit ()
Efeito Secundário	Recebe uma linha do stdin

Nome da função	read_control
Inputs	fi : in_channel
Descrição	Lê uma instrução do fi (control.txt) e chama o read_command
Outputs	unit ()
Efeito Secundário	Nenhum efeito secundário.

Nome da função	controller
Inputs	unit ()
Descrição	Se o simulador está em modo debug chama o read_terminal, se não está em modo debug chama o read_control
Outputs	unit ()
Efeito Secundário	Nenhum efeito secundário.

Variáveis

Nome da Variável	fc
Descrição	Canal de input usado para ler do ficheiro “control.txt”
Tipo	in_channel

Nome da Variável	buffercommand
Descrição	Caracter usado quando é necessário guardar em buffer uma instrução. Quando não há nada em buffer tem armazenado o dollar sign ($).
Tipo	char

Lib

Funções

Nome da função	remove_CR
Inputs	str : string
Descrição	Recebe uma string e remove o último caracter se esse caracter for '\r'. Devolve a string sem esse caracter.
Outputs	string
Efeito Secundário	Nenhum efeito secundário.

Types

Nome do Type	running_state	
Descrição	Type que vai conter os dados dos processos em execução.	
Variáveis	ind : int	Índice da tabela de pcbs (pcb_tabela) do processo que está a correr.
	pid : int	Pid do processo que está a correr.
	pc : int	Pc do processo que está a correr.

Nome do Type	instruction	
Descrição	Type que vai ser usado para construir a memória.	
Variáveis	ins: char	Caracter usado para representar o tipo de instrução.
	n : int	Número usado para o caso da instrução ter um inteiro.
	name : string	String usada o caso da instrução ter uma string.

Nome do Type	newP	
Descrição	Type que contém as informações lidas de uma linha do plano.txt.	
Variáveis	name : string	String com o nome do programa
	time : int	Int que contém o tempo a que o processo deve ser criado e inserido na ready queue
	priority : int	Int que contém a prioridade do processo

Nome do Type	pcb	
Descrição	Type que contém os dados necessários para construir um processo.	
Variáveis	name : string	String com o nome do processo
	start : int	Int com o primeiro endereço que o processo está em memória.
	variable : int	Int que contém o valor da variável do processo.
	pid : int	Int que contém o pid do processo
	ppid : int	Int que contém o ppid do processo
	priority : int	Int que contém a prioridade do processo
	arrival_time : int	Int que contém o tempo de chegada do processo
	time : int	Int que contém o tempo que o processo esteve no CPU
	pc : int	Int que contém o program counter do processo
	status : int	Int que indica o status do processo | 0 -> ready | 1 -> running | 2 -> blocked | 3 -> terminated
	finish : int	Int que contém o tempo a que o processo terminou

Nome do Type	pcb_list
Descrição	Type que serve de construtor para a pcb_list

Variáveis

Nome da Variável	memory
Descrição	Array que serve de memória para o nosso simulador.
Tipo	instruction array

Nome da Variável	next_memory_index
Descrição	Inteiro usado para guardar a próxima posição livre da memória.
Tipo	int ref

Nome da Variável	clock_flag
Descrição	Boolean usado para terminarmos o programa .
Tipo	bool ref

Nome da Variável	executing_flag
Descrição	Boolean usado para determinarmos quando devemos usar a função execute.
Tipo	bool ref

Nome da Variável	rem_time
Descrição	Inteiro usado para determinarmos quanto tempo um programa deve ser executado
Tipo	int ref

Nome da Variável	time_flag
Descrição	Boolean usado para auxiliar na contagem do tempo.
Tipo	bool ref

Nome da Variável	debug_mode
Descrição	Boolean usado para determinar se o utilizador pediu para entrar no modo de debug.
Tipo	bool ref

Nome da Variável	time_flag
Descrição	Boolean usado para auxiliar na contagem do tempo.
Tipo	bool ref

Nome da Variável	time
Descrição	Inteiro usado para contar o tempo.
Tipo	int ref

Nome da Variável	time_quantum
Descrição	Inteiro usado para determinar o time_quantum do simulador (tempo máximo que um processo deve ser executado).
Tipo	int ref

Nome da Variável	next_pid
Descrição	Inteiro usado para atribuir aos processos pids diferentes.
Tipo	int ref

Nome da Variável	pcb_table
Descrição	Lista que contém todos os processos que são criados quando o simulador está a correr.
Tipo	pcb list ref

Nome da Variável	newQ
Descrição	Queue que contém todas as informações que são lidas do plano.txt para, no tempo devido, serem criados processos.
Tipo	newP Queue.t

Nome da Variável	readyQ
Descrição	Queue que contém todos os processos que estão prontos a executar.
Tipo	pcb Queue.t

Nome da Variável	blockedQ
Descrição	Queue que contém todos os processos que foram bloqueados durante a execução do simulador.
Tipo	pcb Queue.t

Nome da Variável	terminatedQ
Descrição	Queue que contém todos os processos que terminaram durante a execução do simulador.
Tipo	pcb Queue.t

Nome da Variável	runnning_proc
Descrição	Variável que contém as informações do processo em execução.
Tipo	running_state

Nome da Variável	heap_f
Descrição	Array que contém as solicitações de memória dos processos que foram alocadas pelo algoritmo “first fit” .
Tipo	int array ref

Nome da Variável	heap_n
Descrição	Array que contém as solicitações de memória dos processos que foram alocadas pelo algoritmo “next fit” .
Tipo	int array ref

Nome da Variável	heap_b
Descrição	Array que contém as solicitações de memória dos processos que foram alocadas pelo algoritmo “best fit” .
Tipo	int array ref

Nome da Variável	heap_w
Descrição	Array que contém as solicitações de memória dos processos que foram alocadas pelo algoritmo “worst fit” .
Tipo	int array ref

Nome da Variável	time_list_f
Descrição	Lista que contém os tempos que todas as solicitações demoraram a ser alocadas pelo algoritmo “first fit”. Se uma solicitação não for alocada então não é adicionado nada à lista.
Tipo	int list ref

Nome da Variável	time_list_n
Descrição	Lista que contém os tempos que todas as solicitações demoraram a ser alocadas pelo algoritmo “next fit”. Se uma solicitação não for alocada então não é adicionado nada à lista.
Tipo	int list ref

Nome da Variável	time_list_b
Descrição	Lista que contém os tempos que todas as solicitações demoraram a ser alocadas pelo algoritmo “best fit”. Se uma solicitação não for alocada então não é adicionado nada à lista.
Tipo	int list ref

Nome da Variável	time_list_w
Descrição	Lista que contém os tempos que todas as solicitações demoraram a ser alocadas pelo algoritmo “worst fit”. Se uma solicitação não for alocada então não é adicionado nada à lista.
Tipo	int list ref

Nome da Variável	rr_flag
Descrição	Boolean que serve para dizer ao programa que o algoritmo em execução é o round robin.
Tipo	bool ref

Nome da Variável	stop_time_flag
Descrição	Boolean que serve para dizer ao programa para não aumentar o tempo.
Tipo	bool ref

Nome da Variável	preempt_flag
Descrição	Boolean que serve para dizer ao programa que o algoritmo de escalonamento a executar é preemptivo.
Tipo	bool ref

Short

Funções

Nome da função	findProcInd
Inputs	queue : pcb list pid : int count : int.
Descrição	Devolve o índice da pcb_tabela do processo que tem o pid igual ao pid recebido.
Outputs	int
Efeito Secundário	Nenhum efeito secundário.

Nome da função	fcfs
Inputs	() : unit
Descrição	Se não existir nenhum processo a ser executado (running_proc.ind = -1) então tira um processo da readyQ, atualiza o estado do mesmo e atualiza as informações do running_proc.
Outputs	() : unit
Efeito Secundário	Tira um processo da readyQ.

Nome da função	short_sched
Inputs	() : unit
Descrição	Função que executa o escalonamento de curto prazo escolhido pelo utilizador.
Outputs	() : unit
Efeito Secundário	Depende da variável selected_scheduller.

Nome da função	findHighestPriority
Inputs	pri : int
Descrição	Devolve o processo com a maior prioridade na Ready Queue.
Outputs	pcb
Efeito Secundário	Nenhum efeito secundário.

Nome da função	priority_p
Inputs	() : unit
Descrição	Função que executa o algoritmo de prioridade com preempção.
Outputs	() : unit
Efeito Secundário	Depende da prioridade dos processos prontos a serem executados.

Nome da função	priority
Inputs	() : unit
Descrição	Função que executa o algoritmo de prioridade sem preempção.
Outputs	() : unit
Efeito Secundário	Depende da prioridade dos processos prontos a serem executados.

Nome da função	round_robin
Inputs	() : unit
Descrição	Função que executa o algoritmo Round Robin.
Outputs	() : unit
Efeito Secundário	Depende do time quantum.

Nome da função	findShortestJob
Inputs	burst : int
Descrição	Devolve o processo com o menor Burst Time.
Outputs	pcb
Efeito Secundário	Nenhum efeito secundário.

Nome da função	sjf_p
Inputs	() : unit
Descrição	Função que executa o algoritmo Shortest Job First com preempção.
Outputs	() : unit
Efeito Secundário	Depende do burst time dos processos.

Nome da função	sjf
Inputs	() : unit
Descrição	Função que executa o algoritmo Shortest Job First sem preempção.
Outputs	() : unit
Efeito Secundário	Depende do burst time dos processos.

Variáveis

Nome da Variável	selected_scheduller
Descrição	Inteiro usado para guardar a escolha do escalonamento de curto prazo do utilizador
Tipo	int ref

Memory

Funções

Nome da função	deallocate_mem
Inputs	pid : int
Descrição	Função que serve para desalocar memória de um dado processo.
Outputs	int
Efeito Secundário	Nenhum efeito secundário.

Nome da função	fragment_count
Inputs	heap : int array
Descrição	Função que devolve o número de fragmentos externos.
Outputs	int
Efeito Secundário	Nenhum efeito secundário.

Nome da função	has_memory_available
Inputs	i : int ; n : int ; heap : int array
Descrição	Função que verifica se existe espaço suficiente para um dado processo ou não.
Outputs	bool
Efeito Secundário	Nenhum efeito secundário

Nome da função	first_allocate
Inputs	n : int pid : int
Descrição	Função que executa o algoritmo de gestão de memória First-Fit.
Outputs	int
Efeito Secundário	Nenhum efeito secundário.

Nome da função	next_allocate
Inputs	n : int pid : int
Descrição	Função que executa o algoritmo de gestão de memória Next-Fit.
Outputs	int
Efeito Secundário	Nenhum efeito secundário.

Nome da função	best_allocate
Inputs	n : int pid : int
Descrição	Função que executa o algoritmo de gestão de memória Best-Fit.
Outputs	int
Efeito Secundário	Nenhum efeito secundário.

Nome da função	worst_allocate
Inputs	n : int pid : int
Descrição	Função que executa o algoritmo de gestão de memória Worst-Fit.
Outputs	int
Efeito Secundário	Nenhum efeito secundário.

Nome da função	n_generator
Inputs	() : unit
Descrição	Função que gera aleatoriamente um número.
Outputs	int
Efeito Secundário	Nenhum efeito secundário.

Nome da função	solicitate_allocation
Inputs	pid : int
Descrição	Função que gera solicitações para alocar memória.
Outputs	() : unit
Efeito Secundário	Nenhum efeito secundário.

Variáveis

Nome da Variável	next_fit_index
Descrição	Variável que serve para armazenar o espaço de memória livre seguinte à última alocação.
Tipo	int ref

Ficheiros

Makefile

Ficheiro usado pelo comando ‘make’ para compilar todo o código do trabalho num só ficheiro executável. Utiliza também o OCamlMakefile, um projeto opensource no github que adapta o Makefile para poder compilar OCaml: https://github.com/mmottl/ocaml-makefile

control.txt

Ficheiro onde se encontram os comandos para a execução do simulador.

plan.txt

Ficheiro onde se encontra a descrição dos processos a serem criados no simulador. O formato é: “name time_to_create priority” em cada linha.

nome_do_programa.prg

Programas a serem executados pelo simulador. O formato é “ins n/name” (ins : char; n : int; name : string;).

Conclusão

Com o desenvolvimento deste projeto adquirimos variadas competências tanto a nível prático/técnico como ao nível teórico.

As competências adquiridas ao nível prático/técnico foram: o desenvolvimento da capacidade de programar num projeto de elevada dimensão e complexidade, o aprofundamento e aperfeiçoamento da linguagem de programação funcional OCaml e a melhoria da nossa capacidade de criar algoritmos.

Quanto às competências ao nível teórico foram: um melhor entendimento de como realmente são geridos os processos por parte do sistema operativo e dos diversos algoritmos de escalonamento de processos e de gestão de memória.

Dito isto, a equipa de desenvolvimento crê que os objetivos pretendidos para este projeto foram cumpridos.