Homework 4 - Scheduling - S 03

Due Wed., Feb. 12

For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, Unix Internals is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
Homework 4 - Scheduling - S 03

Due Wed., Feb. 12

For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicate what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
Homework 4 - Scheduling - S 03

Due Wed., Feb. 12

For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
Homework 4 - Scheduling - S 03

Due Wed., Feb. 12

For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
Homework 4 - Scheduling - S 03

Due Wed., Feb. 12

For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, Unix Internals is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicate what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
Homework 4 - Scheduling - S 03

Due Wed., Feb. 12

For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
Homework 4 - Scheduling - S 03

Due Wed., Feb. 12

For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicate what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
Homework 4 - Scheduling - S 03

Due Wed., Feb. 12

For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
Homework 4 - Scheduling - S 03

Due Wed., Feb. 12

For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
Homework 4 - Scheduling - S 03

Due Wed., Feb. 12

For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
Homework 4 - Scheduling - S 03

Due Wed., Feb. 12

For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, *Unix Internals* is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.
Homework 4 - Scheduling - S 03

Due Wed., Feb. 12

For the questions that refer to the SVR4 and Solaris schedulers, an excerpt from Vahalia, Unix Internals is on reserve for this course at Boatwright library - it should be available on e-reserve by the end of this week.

1. Explain how you would set up the SVR4 Unix dispatcher parameter table to tune a system that was used exclusively for batch-style execution of computationally intensive processes. (This is the way most supercomputers are used.)

2. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing the source code? Could such a measure be determined at run time?

3. Imagine a scheduling system that allowed high priority processes to “put in a good word” for the next process they would like to see execute. Suggest a situation where such a capability would be advantageous.

4. Suppose you were writing a scheduling algorithm for an operating system that supported a system call that allowed a process to indicated what kind of activity it expected to be involved in over the near term. The process passes a parameter that indicates the type of activity. What values would be useful for this flag (i.e. what kinds of expected activity should the process be able to notify the system of)? And how would you design the scheduler to take advantage of this information?

5. Suppose a process releases a resource that several other processes are waiting on. When is it better to wake up all of the waiting processes and when is it better to wake just one of them? How would you choose which process to wake up in the latter case?

6. Why do we say that the traditional Unix scheduler doesn’t scale well? How do the changes in the SVR4 scheduler improve this?

7. Solaris 2.x supports symmetric multiprocessing. Explain how the scheduler was modified for this new functionality.